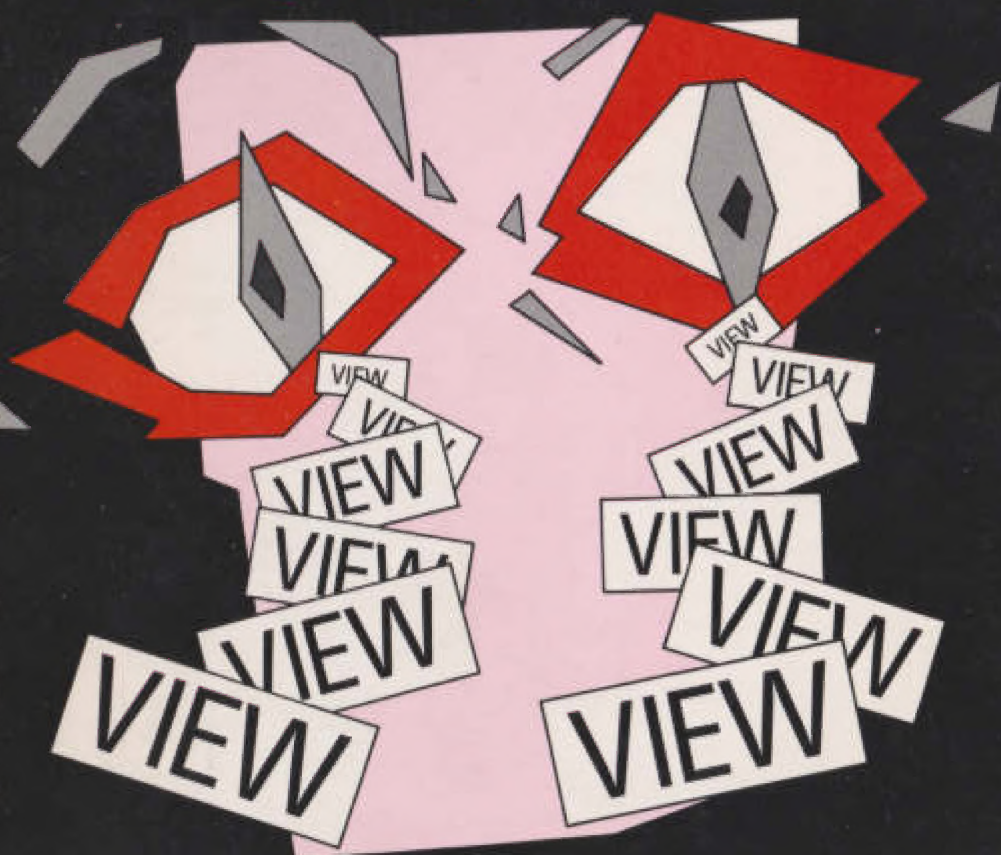

A Dabhand Guide

BRUCE SMITH

VIEW

Including ViewSpell and ViewIndex



DABS
PRESS



VIEW

A Dabhand Guide

Bruce Smith

DABS
PRESS

To Marc Christopher Bruce

(C) Bruce Smith 1987

ISBN 0-870336-00-3

First edition April 1987

The software packages VIEW, ViewSheet, ViewStore, ViewIndex, ViewPlot, ViewSpell, VIEW Printer Driver Generator and OverView are published by Acorn Computers Ltd, under the brand name Acornsoft. Acornsoft is a trade mark of Acorn Computers Ltd, Fulbourn Road, Cherry Hinton, Cambridge England. MacAuthor is published by Icon Technology Ltd, Leicester England. The Sunday Times is published by News International PLC, London England. The Apple Macintosh and Laserwriter are produced by Apple Computer Inc.

Within this book the letters 'BBC' refer to the British Broadcasting Corporation. The terms 'BBC Micro', 'Master 128' and 'Master Compact' refer to the computers manufactured by Acorn Computers Ltd under licence from the BBC. The terms 'Econet' and 'Tube' are registered trademarks of Acorn Computers Ltd.

All rights reserved. No part of this book (except brief passages quoted for critical purposes), or any of the computer programs to which it relates may be reproduced or translated in any form or by any means mechanical electronic or otherwise without the prior written consent of the copyright holder.

Disclaimer : Because neither Dabs Press nor the author have any control over the way the material in this book and accompanying programs disc is used, no warranty is given or should be implied as to the suitability of the advice or programs for any given application. No liability can be accepted for any consequential loss or damage, however caused, arising as a result of using the programs or advice in this book or the accompanying programs disc.

Published by Dabs Press, 76 Gardner Road, Prestwich, Manchester M25 7HU England.

Typeset in 10 on 11 pt Palatino by Dabs Press using the Acornsoft VIEW wordprocessor, MacAuthor, and an Apple Macintosh and Laserwriter. Printed in Great Britain by A. Wheaton and Co Ltd, Exeter, Devon. A member of the BPCC Group.

Contents

Introduction	8
1:The Wordprocessor	11
The equipment	13
A different View	14
Into View	15
A constant View	16
2:Introduction Command and Edit Mode	17
Command Mode	17
The Status Information	18
Edit Mode	20
Entering Text	21
The Roving Cursor	22
The Function Keys	23
Deleting Text	24
The Word Count	25
The BREAK Key	26
3:The Ruler	27
Changing Rulers	28
Previewing Text	29
TAB Stops	30
Printing Text	32
4:Saving and Loading Text	34
Cassette Filing System	34
Disc Filing System	35
Advanced Disc Filing System	35
Network Filing System	36
Loading Text	36
A Quick Save	36
Locking Files	37
Filename Choice	37
Disc Commands	39

VIEWing saved files	40
Bootiful	41
5:Markers	
Moving Text	42
Copying Text	44
Deleting Text	46
Markers 3 to 6	48
Saving Marked Text	49
Loading Text to a Marker	49
Counting with Markers	50
6:Formatting and Justification	51
Insert Mode	51
Justification	52
By Command	54
More on Formatting	55
7:Search and Replace	58
A Folding Case	58
CHANGE	59
REPLACE	61
SEARCH	62
8:Stored Commands and Margins	65
Centred Text	66
Right Justify	67
Line Spacing	68
Margins	69
Beyond the Right Margin	72
Margin Stored Command	72
9:Page Layout	74
The Default Page	74
Headers and Footers	76
Footers	76
Headers	77
On and Off	78
Positioning the Footer	79
Positioning the Header	79
Page Length	79

Page Ejecting	81
Paging and Page Numbers	82
10:Macros	85
Number Registers	88
Chapter Sub-headings	89
11:Printers and Drivers	92
Printer Types	92
The Printer Driver	93
Free Bees	93
Control Codes	94
Up to a Point	95
Underlined Text	96
HomePDG	96
Loading the Driver	97
More Codes	97
Highlight Codes	99
Extended Highlights	100
SHEETS versus PRINT	101
Microspacing	102
Text Formatting	102
Making a Hash of Pounds	103
Listing - HomePDG	104
12:Macros Revisited	108
A Simple Example	108
Chapters, Sections and Sub-headings	109
13:Long Documents	113
Demon Disc	113
The EDIT Input File	114
The EDIT Output File	114
Printing Long Documents	115
14:Hints and Tips	116
Abbreviations	116
Star Commands	117
A Touch of Colour	118
!Boot Files	119

Highlighting Problems	120
Listing - ExtKeys	120
DIY Editing	121
*CONFIGURE	124
Pounds and Hashes	124
15:VIEW Manager	125
ExCat	125
VIEW Manager	127
Template	128
Precis	129
Extending VIEW Manager	131
Listings - Excat	132
- VIEW Manager	134
16:ViewSpell	141
Checking a File	142
The Spell Check	144
Spelling Check	144
Marking the document	145
Disc Managment	147
!Boot Files	148
Markers	148
User Dictionaries	150
Adding Words	151
Indexes	153
17:ViewIndex	154
Stage 1 - Marking Entries	154
Highlighting Problems	156
Stage 2 - Create Intermediate Index	156
Stage 3 - Creating the Main Index	157
Stage 4 - Editing the Index	159
Indexing by Sections	160
18:VIEW Utilities	161
Utility 1 - The Screen Layout Driver	161
The Preview Program	162
A Stationery Difference	162
Early View	164
Utility 2 - The VIEW Recovery Program	164
Dry Run	165

Utility 3 - The File Checker	169
Utility 4 - The Highlight Matcher	169
Utility 5 - The View Help ROM	170
Forming the Image	170
Adding your own	171
Listings - Page Driver	172
- Restore Text	177
- File Checker	179
- Highlight Matcher	182
- Help ROM	187
 19:A Matter of Style	 191
 20:OverView	 193
New Commands	194
 Appendices	 195
A: Quick Command Guide	196
B: Quick Stored Command Guide	200
C: Quick Highlight Code Guide	203
D: Quick * Command Guide	204
E: Quick Error Guide	207
F: Quick ViewSpell Guide	210
G: Quick ViewIndex Guide	213
H: Priority VIEW	214
I: Increasing your Bytes Free	217
J: Epson Printer Control Codes	219
K: Technical Notes	226
L: VIEW - Version Differences	227
M: The Programs Disc	231
N: Dabhand Guides for your Micro	233
 Index	 235

Introduction

Letters, memos, notes, reports, documents, manuals and books: all need writing, and for most of us this has usually been done with pen and paper. In 1875 Mark Twain broke the ice and set a trend which transformed text preparation.

Twain had had the foresight to invest \$125 in a brand new state-of-the-art Remington typewriter. Using it he prepared the manuscript for a book and submitted it to a publisher. The first ever typed manuscript was accepted, and now millions have enjoyed and delighted in *The Adventures of Tom Sawyer*.

The BBC micro could be considered a latter day equivalent. The steps between handwriting and Twain's typewriter and between typewriters and wordprocessors are of equal magnitude: wordprocessing has revolutionised text preparation the world over.

The VIEW wordprocessor is a good one. Many would call it powerful, but this is not a word I like. However, as someone who has used a variety of wordprocessing packages on a variety of machines, I can vouch for the fact that it is fast, efficient and, despite possible first appearances, easy to learn, certainly it is much more sophisticated but easier to use than wordprocessors written for more 'powerful' microcomputers.

You can use your wordprocessor for all of the tasks mentioned in the first line and many more - the complexity of text has no bearing. The first simple notes around which this book was built were made in VIEW, and of course the 50,000 words of the actual text were entered and edited in VIEW.

At first sight VIEW may seem a bit cold and unfriendly, but with some time, some practice and a copy of this Dabhand Guide, it will become a useful and easy tool. I have tried to make VIEW: A Dabhand Guide a practical book which uses and asks you to follow plenty of examples. However that does not make it a one-off read - far from it. It will also provide you with the only complete source of reference on VIEW, and its many examples will provide you with the template to apply to your current task.

In addition there are numerous utility programs which are written for practical use and these and many others can be found on the Programs Disc (see Appendix M).

VIEW and *VIEW: A Dabhand Guide* could be just about the best match since Tom Sawyer and Huck Finn.

Bruce Smith

Barnet. February 1987

Acknowledgements

There are many people to thank for their help and preparation of this volume. Paul Horrell for editing and commenting on the text, David Atherton for proof reading and supplying one or two programs, Graham Bell for his superb page driver program, VIEW spooler on the Programs Disc and general hints and tips, Graham Bartram for designing and setting the text on his Macintosh DTP system, Paul Holmes for a catching cover design, The Times and Sunday Times for permission to use the LUNAR text, Rob MacMillan for supplying information (eventually), Sue Wall and Richard Morris at Acorn for their support, Lynda, "01", and Roger, Debbie and Jim Day at BBC Soft for putting up with him during the transition(!). And finally Tessie, Sarah and Marc.

Dedication

This book is dedicated to Marc Christopher Bruce aged 9 months.

Thank you ...

...for buying /borrowing/stealing this book, we hope you have enjoyed it! If you have any comments or suggestions to make about this Dabhand Guide then we'd love to hear from you. Please address all correspondence to Dabs Press at the address on page 2. Correspondents and mail order purchasers will be automatically advised of future Dabs releases, unless they request otherwise. Personal details held will be provided on request in accordance with the Data Protection Act.

Conventions

The following type face conventions have been used in this Dabhand Guide:

Section Headings look like this

Sub Sections in this style

This is the normal text

Text in this typeface is to type in
Program listings look like this

Anything in this typeface is an actual key ie SHIFT

Any listings appearing in a Chapter can be found at the end of the chapter in question.

Note that the | symbol shown in various places appears on the BBC Micro with a break in the middle (Modes 0-6) or as two vertical bars (Mode 7).

About this book

This book was completely written on the VIEW 3 wordprocessor, using a Master Compact. It was then transferred to an Apple Macintosh using Modem Master by BBC Soft (on a Master 128) and Red Ryder on the Macintosh. The text was then formatted for output using MacAuthor, one of the few British desktop publishing packages for the Macintosh. Finally it was printed on an Apple Laserwriter Plus, and passed as camera-ready copy to the printer, A Wheaton and Co. The book cover was done by conventional means.

About Dabs Press

This is the first book from Dabs Press, a new software and book publishing firm. In Appendix N you will see our publishing plans for 1987. If you have any comments or suggestions about future books, or perhaps you would like to write a book or software pack for us, please contact David Atherton at the address on page 2.

1: The Wordprocessor

Alloy Aluminum Ltd, Ally Pally Road, Alingham

Mr B Keeper,
103 Acacia Avenue,
Arkley,
Herts.

30 September 1987

Dear Mr Keeper,

Because of the expansion of our company, we are considering opening a showroom in the ARKLEY area. To help promote our company in this area, we are looking for a number of homes into which we can install our full and comprehensive range of windows, doors and patio doors.

This is where you come in, Mr Keeper. We feel that 103 Acacia Avenue would be an ideal site for just such a showroom. We would wish to display an advertising board at 103 Acacia Avenue for a short period and in addition take a few photographs showing the improvements to the house.

In return for this, Mr Keeper, we are prepared to subsidise the cost of installation or, if you prefer, pay you handsomely in CASH.

But do hurry, Mr Keeper, because we need just one or two more properties such as 103 Acacia Avenue to complete our collection of showrooms. Contact me straight away at the office phone number. I look forward to chatting to you in the near future.

Yours sincerely,

Connie Merchant

Connie Merchant
Marketing Director

How many times have you received a letter along similar lines? It will usually be neatly printed and very personal, using your name and address several times within the body of the letter. You also know that, although you may be the only person in your immediate area to have received such a letter, thousands of other people all over the country have also had exactly the same one, but with their details inserted in place of yours.

How are these letters mass produced? It's unlikely that a pool of typists has been employed to type each letter individually, inserting your name and address where appropriate. Of course not - these letters are produced on a wordprocessor, and possibly a BBC micro using the VIEW wordprocessing program.

The mass production of personalised letters is one of the functions of a wordprocessor. Text, typed in at the keyboard, is stored in the computer's memory. The text is displayed on the screen which is, in effect, a sheet of electronic typing paper. Names, words and addresses can be changed automatically with the absolute minimum of fuss.

The letter above shows another feature that distinguishes wordprocessing from traditional typing. The last characters on each line form a straight margin, with no ragged edges. I deliberately made the VIEW wordprocessor do this - I could equally easily have specified it not to be done, and thereby produced a ragged edge. I think this justification, as it is called, gives the whole letter an air of professionalism.

Once a letter or some text has been typed in (from now on, we'll use the words 'document' or 'text' to cover all possibilities), you can save it on a cassette, disc, network - whichever you have. This allows the text to be retrieved at a later date - tomorrow, next week, next month or even next year - so that it can be altered as required, a process known as editing, and then printed on to paper. Using a traditional typewriter the whole lot would have to be re-typed each time: a waste of time and effort, especially if the document is several pages long.

If you want several copies of your document then simply print them: a copy for you, a copy for your file and a copy to the person concerned (get rid of the photocopier!). The whole point is that VIEW allows you convert your Beeb into a sophisticated word manipulator, and wordprocessing is all about flexibility.

The Equipment

Let's look at what you're going to need to get your VIEW wordprocessing station into action. You probably have most of the equipment already - particularly Master Compact owners, who can skip the next few paragraphs. Obviously you will need the BBC micro and a monitor. A television is quite adequate for most purposes, though it does have some minor drawbacks compared with a specially designed computer monitor. I use a medium resolution colour monitor and find it fairly easy on the eyes for prolonged use. Many people, though, prefer a green screen monitor. The choice is up to you.

To save your documents you will need either a cassette recorder or a disc drive. Both perform admirably the task of loading and saving documents, but there's no doubt that a disc drive is much quicker. A large VIEW file takes under two seconds to load from disc compared with three or four minutes from tape. The first book I wrote using a wordprocessor in 1982 was 40,000 words long and had to be saved on several cassette tapes!

If you are using VIEW for serious applications, such as mailing lists or for your business, then a disc drive would be much more efficient; I'd say that if you are serious about your wordprocessing then a disc drive is a must. Any BBC micro-compatible disc drive will work with VIEW - no extra programs are necessary. If you are unsure about which disc drive to buy, I recommend that you look at the magazine press for advertisements and, more importantly, reviews. I wrote two such reviews for the July 1985 and February 1986 issues of *Acorn User*. Obviously, new disc drives may well be on the market when you read this, but whatever brand name you buy I would strongly recommend that you go for a 40/80 track switchable drive. This will give you the best value for money and the most memory for your documents.

If you have access to a network, such as Acorn's Econet, this can be used to save documents. Generally a network will use disc drives or a Winchester hard disc to act as its storage medium. As a user of the network this need not concern you, but if you're interested in how it works, ask your network manager to demonstrate and explain the various components of the network.

A printer is a highly desirable item. You will need to print your documents onto paper, producing what is often referred to as the 'hard' copy, so that you can give them to other people. However, for limited use a printer may not be essential: you can save and load your

text on tape or disc, so as long as the person you're sending your document to has VIEW installed in their Beeb they can load it in and read it.

Which printer to choose will depend on what you want it for. Daisywheel printers have very professional typefaces and are particularly suited for business. Good printers of this sort can be quite expensive, but they also allow you to change the typeface simply by inserting a new character font, just as you would on a standard golf-ball typewriter.

If you wish to produce text that contains features such as underlining, italics, emphasised print, double print and perhaps graphics dumps (facsimiles of screen images) then you should look out for a dot-matrix machine. These also tend to be reasonably priced, and happen to be my favourite type. The most common dot-matrix printer is the Epson, and most other printers tend to be Epson-compatible. Virtually all Epson-compatible printers offer a feature known as NLQ, which stands for 'near letter quality' and allows you to print text to a quality approaching that of a daisywheel printer.

If money is no object then a laser printer is the ultimate to aim for. But in every case, read the magazine advertisements and reviews before you buy.

How you arrange your VIEW wordprocessing station is up to you, and largely depends on the equipment you are using. The most important thing, though, is that it's comfortable to use. The table or desk it is housed on must be of an adequate size to allow all the kit to be arranged neatly, with ample space left over for pen, paper, telephone and coffee. The monitor should not strain your neck by being at the wrong height: ideally you should be looking slightly down at it. Disc drives should be easily accessible, as should the printer, though this does not have to be at arm's reach.

Network users will probably find that there is a special printing station - often referred to as the printer server - elsewhere in the building, where documents can be printed out away from your own personal station. Again, ask your network manager where it is and how to operate it.

A Different VIEW

There are several different versions of VIEW around. Fundamentally they all perform the same function, but later versions have several

small programming oddities/errors (called 'bugs' or features!) removed and many other small enhancements. Master 128 and Master Compact owners have VIEW supplied with the micro and these versions are the recent releases. BBC model B and B+ owners have had to buy VIEW separately and there are three releases available:

VIEW 1.4 VIEW 2.1a VIEW 3.0

VIEW 1.4 is the earliest release and VIEW 3 the latest. The full list of differences between the various versions can be found in Appendix L. Most of this book is relevant to any version - however it is worth referring to the Appendix just in case.

To find out which version of VIEW you have, type the following into your micro:

*HELP

and press the RETURN key. A list of the ROM chips in your computer will be displayed (you may need to press the SHIFT key to see the complete list). Look for the word VIEW. After this will be a number, which is the version you have. Make a mental note of it.

Into VIEW

The way you get into VIEW depends on which micro you have. BBC model B and B+ owners will, the first time they use VIEW on their machine, have to install the ROM chip into the micro. The VIEW pack contains instructions on how to do this - if in doubt consult your local Acorn-approved dealer or computer club. With the chip installed all you need to type is:

*WORD <RETURN>

to take you into the VIEW wordprocessor.

Master 128 owners already have the VIEW ROM in their micro. So simply type:

*WORD <RETURN>

as described above.

Master Compact owners have VIEW stored on the Welcome disc, and there are two ways in which this can be loaded into the Compact. The quickest is simply to put the Welcome disc into the drive and type:

*MOUNT <RETURN>

*WORD <RETURN>

The disc will show signs of life and within a few seconds you will have VIEW installed and ready to use. A second, but more long winded, approach is simply to 'Boot-up' the Welcome disc to get the Front End and select VIEW from the Applications Menu.

VIEW is now ready for the preparation of letters, reports, memos and even more adventurous tasks such as the writing of books - this book was written entirely on the VIEW wordprocessor. The display that first appears on your screen (a displayed page is known as a screen) is called the Command Mode screen - one of two screens that VIEW generates, the second being the Edit Mode screen. We'll examine each in the next chapter.

A Constant VIEW

If you plan to use your computer exclusively for VIEW and you are using any micro other than a Master Compact, then it is possible to make the computer go straight into VIEW when you switch on. Appendix H shows how to arrange this.

2 : The Command & Edit Modes

Command Mode

With VIEW installed and selected you are in what is called the Command mode. On the monitor you should have a blank screen with the word VIEW in the top left-hand corner, followed by these lines of information:

```
Bytes free xxxxx
Editing No file
Screen mode x
```

The x's represent numbers which will vary according to the micro you are using. There may be an extra line of information beneath these, saying either 'Printer default' or 'Printer Epson' (on the Compact). All this is collectively referred to as 'Status information'. It provides useful on hand details about the current state of VIEW in your system. Before we discuss what it means look a few lines further down the screen, where you should see the VIEW prompt which will look like this:

```
=>_
```

The horizontal line will be flashing: this is the cursor. Type the following text at the keyboard - as you type it will appear after the prompt. Press the RETURN key after the 3:

```
MODE 3
```

The screen's effective size will change - the status information will appear somewhat smaller and you will find that the numbers presented by the status information will also alter. We have in fact entered into the system a VIEW command, and for this reason the screen presentation is called the Command mode. We are here discussing two uses of the word 'mode', so be careful to distinguish between the Command and Edit modes of VIEW and the screen modes of the micro. If you are vaguely familiar with BBC BASIC or have a working knowledge of your micro you will be aware that it supports a number of screen modes. A screen mode simply determines how much graphic or textual information can be displayed on a screen in one go.

Mode 3 is a particularly good mode in which to operate a wordprocessor, as it allows up to 80 characters to be displayed across the screen in 25 rows at any one time. In all, VIEW supports Modes 0 through to 7. To see the difference type the following command:

MODE 7 <RETURN>

The quantity of text that can be displayed becomes somewhat smaller - the status lines occupy about one-eighth of the screen. Now type:

MODE 5 <RETURN>

The status information now occupies about half the screen because the text is made larger. Choice of mode is subjective; some people prefer mode 0, but I normally work in Mode 3 and so the examples based in this book all assume a Mode 3 screen.

The Status Information

As you were changing modes, you may have noticed that the 'Bytes free' figure changed. This is because the screen uses prospective text storage memory to work. Some modes use more memory than others, and the more memory required by the screen, the less is available for your text in VIEW. The bytes free value is the amount of memory within the computer that is available for text. You can think of bytes free as being characters remaining. A byte is required to store each character of the text you type in at the keyboard or load from tape or disc, so if the bytes free value was 12000, you would have room to enter about 12000 characters. It is probably quite difficult for you to imagine 12000 characters - on average a word will contain about 6 characters, so the number of words you can enter with 12000 bytes free is arrived at by calculating the simple sum:

$12000/6 = 2000$ words

As this book contains around 400 words of text per page, that's enough room to store text to fill five pages of this book. Table 2.1 shows the amount of memory available for the various BBC micros in different configurations. The figures are a rough guide, so don't be concerned if they differ a little from your status information.

Appendix I contains more details as well information on how to get more memory should you find yourself running low. Modes 128 to 135 are 'Shadow' modes available to Master 128 and Master Compact owners. These modes can be added to a standard BBC B; again, see Appendix I. B+ owners do have shadow modes available and in

general the Master 128 figures minus about 4000 bytes (around 600 words) apply.

Master 128 and Compact	Mode	Bytes Free	Words
	128 to 135	28,926	4100
	7	27,902	3900
	6	20,734	2900
	5	18,686	2650
	4	18,686	2650
	3	12,542	1800
	2	8,446	1200
	1	8,446	1200
	0	8,446	1200
BBC B	Mode	Bytes Free	Words
	7	24,318	3500
	6	17,150	2450
	5	15,102	2150
	4	15,102	2150
	3	8,958	1300
	2	4,862	700
	1	4,862	700
	0	4,862	700

Table 2.1. Text storage comparisons.

'Editing no file' - unsurprisingly, this status line tells you that you are currently editing no file. When you have reached the stage of entering text and saving it on disc, subsequent reloading of the text will result in the 'no file' being replaced by the file name of the text. Thus if the text was saved as 'Chap1', loading this back would cause the Editing status line to show:

Editing Chap1

'Screen mode' should be self explanatory - it indicates the current screen mode you have chosen.

The final 'printer' line is somewhat more complicated. To allow you to produce special effects such as underlined or italic text when you print out a document, a program called a printer driver must be loaded into VIEW. The printer driver sends special codes to the printer to enable it to produce the effects. This final status line indicates the type of printer driver you are using.

The subject of printer drivers is often a source of headaches for many users new to VIEW, so a whole chapter is devoted to the subject later on. Not having a printer driver does not mean we cannot print text on the printer - far from it. But we can't at this stage get those fancy effects.

Enter the following command after the VIEW prompt:

NEU

Remember to press the RETURN key at the end. This command sets VIEW up ready to receive some text, so let's do so.

Edit Mode

Press the ESCAPE key. The Command mode screen disappears and drops you into another, almost blank, screen - this is the Edit mode screen, which is where all your wordprocessing is done. The screen is not entirely blank: there are two lines of characters - the line at the very top contains a couple of letters followed by a few of dots and asterisks, looking like figure 2.1.

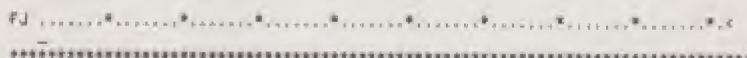


Figure 2.1. A standard VIEW Edit screen.

This is the ruler, and it has a very important role in VIEW. It determines two main things. First, how wide your text will be when displayed on screen or printed out, and second, where the TAB stops are positioned. The concept of the TAB is very similar to its use on a conventional typewriter. The letters to the extreme left of the ruler will either be FJ (for Master, and BBC Models B and B+) or FI (on a

Master Compact). As the ruler is so important, it is the subject of the next chapter.

The lower of the two lines is a line of asterisks and this simply marks the end of your text - the bottom of your sheet of electronic paper.

Press the ESCAPE key once more. You move back into Command mode. Press ESCAPE again and you're back in Edit mode. The ESCAPE key is our mode selection key, then: it allows us to 'toggle' between VIEW's two modes of operation.

Entering text

As text is entered into your wordprocessor onto the Edit mode screen, first check that you are in Edit mode - press ESCAPE if not.

The flashing cursor shows where text will appear as you type at the keyboard. Ensure that the CAPS LOCK and SHIFT LOCK lights on the keyboard are extinguished. If either are lit then simply press the appropriate key on the lower right hand side of the keyboard.

Now type the following:

The VIEW wordprocessor

Figure 2.2 shows how the screen should look.

The VIEW wordprocessor.

Figure 2.2. Entering text into VIEW.

The upper-case (capital) characters are obtained in the normal way, by pressing the SHIFT key while hitting the letter required on the keyboard. The flashing cursor moves along in front of each letter as you type - it should now be sitting after the 'r' at the end of 'wordprocessor'. To move onto a new line, so that we can add more text below what we have just typed, it is only necessary to press the RETURN key. Do this and see how the line of asterisks at the end of text moves down, leaving the cursor at the start of a new blank line. Type the following onto this line:

For BBC B, B+, Master and Compact micros.

Press RETURN at the end of the line and the screen should look as figure 2.3.



1

The SHIFT key can also be used in conjunction with the arrow keys. Pressing SHIFT with either of the left- or right-arrow keys will move the cursor to the first letter of the next word in that direction. This technique is useful when you wish to move rapidly to a word somewhere within a line of text. Pressing the SHIFT key with either of the up- or down-arrow keys will cause the cursor to move 24 lines in the appropriate direction. At the moment we have only a couple of lines of text, so this one cannot be tried out yet.

The Function Keys

If you haven't already done so, find your VIEW function key strip and place it above the red function keys of your computer. On the BBC models B and B+ and Master Compact, it will be a single strip of laminated paper. The Master 128 key strip is a four way affair, one of which will be marked with VIEW: place this facing upright. The function key strip looks complicated, but it isn't really so bad. It's just that, with the exception of f9, all keys have three functions.

The function keys have three levels of use in Edit mode. The first level is obtained simply by pressing the function key - the effect it will have is stated on in the box immediately above the function key. For example pressing function key f9 will DELETE CHARACTER, ie, delete or erase the single character above the cursor.

The second level of use is when a function key is pressed while holding the SHIFT key down. Here, the effect is documented in the middle row of the keystrip, labelled SHIFT at the extreme left hand side. Pressing SHIFT-f0 will MOVE BLOCK - that is move a block of text from its current position to the cursor position.

The final level of function key use is obtained by pressing the function key with the CTRL key held down. The action of the CTRL and the function key is described in the top row of the keystrip. Thus pressing CTRL-f0 will DELETE BLOCK - delete or erase a block of the document.

A word of warning. Several of the function key commands can delete text, and unfortunately the keys that are used to allow you to insert text are side by side with these. For example, hitting f7 instead of f6 will cause you to delete a line of text when you really wish to add one. So, until you are very familiar with the use of the function keys, always double check the key you are going to press before you actually press it!

Four function keys allow us to move the edit cursor around the text in a similar fashion to that described for use of arrow keys with the CTRL key. The keys in question and their operation are as follows:

- f1 - moves cursor to top of text
- f2 - moves cursor to bottom of text
- f5 - moves cursor to end of line
- f6 - moves cursor to start of line

Try experimenting with these within Edit mode on the text already typed in. The equivalent CTRL key movement for each is listed below:

- f1 = CTRL-↑
f2 = CTRL-↓
f5 = CTRL-→
f6 = CTRL-←

Deleting Text

Now that we have typed some text let's see how we can delete and edit what we have written so far. Move the cursor so that it is under, the T in 'The', best done by pressing CTRL-↑.

To delete the character that sits directly above the cursor we press function key f9 - to remove the word 'The', press it three times. Each time you press f9 the letter above it is erased and the rest of the text on the line moves a character to the left to fill the gap left by the deleted letter. The result is as shown in figure 2.4.

```

FJ *****
_U1EH wordprocessor
For BBC B, B+, Master and Compact micro
*****

```

Figure 2.4. Deleting a word in VIEW.

Our next task is to insert an 'A' so that the first line of our small document will read

A VIEW Wordprocessor

Before we can insert an 'A' we must make a space for it - this is done by pressing function key **f8** whose action is **INSERT CHARACTER**. Press **f8** now. The text will move one space to the left, creating the gap into which you can type an A (remember upper case characters

are obtained with the help of the SHIFT key). The result is illustrated in Figure 2.5.

```
F) .....^.....^.....^.....^.....^.....^.....^.....^.....^.....  
A-VIEW wordprocessor  
For BBC B+, Master and Compact micros  
  
*****
```

Figure 2.5. Inserting a letter.

And that is how easy it is to delete and insert text or single letters. Because of this it does not matter how sloppy a typist you are. Despite writing some 18 books, I am still limited to two fingers, although I sometimes use four (two on each hand) if I am not in a great hurry. But often I am anxious to get my thoughts down before I forget them. With a wordprocessor, this is easy as I do not have to worry to much about grammar, spelling or style: the text can be edited once it is written. With a conventional typewriter this would not be possible.

As a simple exercise, try inserting the word 'the' into the second line after 'For' so that the end result looks like figure 2.6.

[illegible]

Figure 2.6. Inserting extra words.

The Word Count

When preparing a document, it can be very useful to know how many words it contains. VIEW supports a command that will provide you with this information. The command is COUNT and to use it you must be in Command mode. With the above text in VIEW, if you press ESCAPE to return to Command mode and type:

COUNT (RETURN)

it will almost immediately respond with the message:

12 word(s) counted.

as shown in figure 2.7.

```
VIEW
Bytes free 12516
Editing no file
Screen mode 3

=>COUNT
12 word(s) counted
=>
```

Figure 2.7. Using COUNT in command mode.

Go back to Edit mode and enter one or two more words, your name perhaps, on the blank third line. Now perform a COUNT and see that the words counted will have increased by the number of words you have entered. VIEW commands can be entered as either lower case or upper case characters, or even a mixture of both.

count	COUNT	Count
-------	-------	-------

will all have the same effect. Throughout this volume the upper case setting is used to help distinguish commands from normal text.

The BREAK key

The BREAK key can affect the text you have held in VIEW so it is most important to become familiar with its action. In versions of VIEW in a BBC B or B+ micro a default screen mode will be displayed - this is the mode which appears when you switch your micro on. Your text will also seem to have disappeared. You will be deposited in Command Mode - so immediately type in the following command:

```
OLD <RETURN>
```

This will restore your text. Do not attempt to enter Edit mode and start typing again, without first typing this command, otherwise your text will be 'lost'. All that remains is to reselect the screen mode that was in use.

In versions of VIEW supplied on a Master 128 or Master Compact the only side effect of pressing BREAK is that you will be transported into Command mode if you were in Edit mode at the time of pressing the BREAK key, but your text will be preserved, so typing OLD is not necessary.

3 : The Ruler

Enter Command mode and clear any text you may have entered by using the NEW command. Ensure that you are in MODE 3, then enter Edit mode and type the following line of text exactly as shown, stopping immediately after the M in 'ROM':

The VIEW wordprocessor runs on all BBC micros. It is supplied as a ROM

NB: Don't type RETURN after the 'It'. The sentence reaches from the left-hand side of the screen over almost to the very right-hand side. The last character on the right-hand side of the ruler is a <. This is the edge of paper marker and shows where the electronic paper's right-hand edge is. What will happen if we continue typing text so that we go beyond the edge of paper marker? Type the word 'chip' and see what happens. Before you have typed the whole word, it moves, as if by magic, onto the next line. VIEW has formatted the text for us. In fact the F in the right hand edge of the ruler means that formatting is enabled - that is switched on.

```

d .*.-----*.-----*.-----*.-----*.<
I BBC micros. It is supplied as a ROM

C micros. It is supplied as a ROM chip. This makes it easy to...
*****
```

Figure 3.1. Typing off screen

We can switch formatting off. To do this, press CTRL-f2, that is, hold down the CTRL key, press function key f2 and then release the CTRL key. Notice that the F in the ruler line has vanished, because formatting has been disabled or switched off. Move onto a new line by pressing the RETURN key and now type the sentence again. This time, when you reach the end of the line the word 'chip' does not move onto the next line but instead stays after 'ROM' and you type off the screen, which moves to the left and will continue to do so as you enter more text, until the main text has completely disappeared - this is illustrated in figure 3.1.

Pressing the RETURN key will get you back onto the main screen on the line below the unformatted line. Turn formatting back on by pressing CTRL-I2. The F will reappear in the ruler line, but the line of text is not automatically formatted on screen as one might expect. We can, however, instruct VIEW to format it to fit onto the screen. Move the cursor to the first character on the line and then press function key f0 - which is labelled FORMAT PARAGRAPH. The line will be formatted and now spreads across two text lines with the result shown in figure 3.2.

```
FJ .....*.....*.....*.....*.....*.....*.....*.....*.....*
The VIEW wordprocessor runs on all BBC micros. It is supplied as a ROM
chip.
The View wordprocessor runs on all BBC micros. It is supplied as a ROM
chip. This makes it easy to
*****
```

Figure 3.2. Formatting text in VIEW.

Changing Rulers

Just like text, we can edit the ruler to a smaller or larger size as we wish. This is very useful; it provides the main way of varying the text width. Enter Command mode and type NEW to clear text currently in VIEW, and return back to Edit mode. Enter the text shown below so that Edit mode will look somewhat similar to figure 3.3.

```
FJ .....*.....*.....*.....*.....*.....*.....*.....*.....*
The VIEW wordprocessor for the BBC B, B+, Master and Compact micros.
*****
```

Figure 3.3. Text with the default ruler.

Move the cursor down two lines by pressing the RETURN key twice. We are now going to insert a ruler into the text and edit so that it will produce text about half the current width when formatted. To get a new ruler for editing, press CTRL-I5 (which will be labelled either DEFAULT RULER or just RULER on your keystrip). The new ruler appears. Now move the cursor onto the ruler and press f9 23 times to delete the first 23 characters in the ruler. The result should look like this:

```
..*.....*.....*.....*.....*.....*.....*
```

ie, 7 asterisks wide. Press RETURN to move off and below the ruler. Note that the ruler at the top of the screen will have changed to match the size of the new ruler. The ruler at the top of the screen is always the current ruler. You can embed as many different sized rulers in a document as you wish, thereby varying the width of text throughout the document. The ruler only affects the text below it up to the end of the document or until a new ruler is reached.

Under the new ruler, type in the same sentence again. This time it will wrap onto the next line at about the time you type the word 'and', giving a screen like figure 3.4.

```
FJ .....  
The VIEW wordprocessor for the BBC B, B*, Master and Compact micros.  
.....  
The VIEW wordprocessor for the BBC B, B*, Master and  
Compact micros...  
.....
```

Figure 3.4. Formatting text with a new ruler.

Play around with rulers for yourself, experimenting with different sizes. Even when you have typed in text after a ruler you can go back to the ruler and edit it further, inserting full stops with the aid of f8 or removing even more of them using f9. But changing a ruler in this way will not have an immediate effect on the text below it: you have to reformat the text by moving the cursor to the first letter in the first word below the newly edited ruler and then press f0 -
FORMAT PARAGRAPH.

Previewing Text

Although we see on screen the text in the manner in which it will appear when finally printed out, the screen is rather cluttered with the new ruler or rulers. We could really do with a means of seeing the screen formatted correctly but without the rulers. And that is indeed possible with a VIEW command. ESCAPE into command mode and then type in:

SCREEN

After pressing RETURN, the text will appear at the bottom of the screen, formatted as in Edit mode but minus rulers. It will stay here until you press the SHIFT key. Keep the SHIFT key depressed until the familiar VIEW prompt reappears. But why should we want to preview text rather than simply examine it after printing it out?

Because previewing saves time and printer paper. Printing text is often a slow process, and if after printing we decided to make a few changes we would need to print it again to ensure the changes are correct. So always preview text with the SCREEN command, and when you are happy about the presentation you can print the document - a subject we shall be coming to shortly.

TAB Stops

No doubt you will have been wondering about the asterisks in the ruler. These are the TAB stops. TAB stops, used with the TAB key, come into their own when we wish to format both text and, in particular, tables of figures. For example, suppose we wish to format four columns of numbers such that column 1 contains the numbers 1 to 10, column two the numbers 11 to 20, column three 21 to 30 and column four 31 to 40. Hence this should be the final layout:

1	11	21	31
2	12	22	32
3	13	23	33
4	14	24	34
5	15	25	35
6	16	26	36
7	17	27	37
8	18	28	38
9	19	29	39
10	20	30	40

Perhaps the most obvious way to get this effect is simply to type spaces between each column of figures. This is fine for small tables, but for large ones it becomes time consuming and boring. The TAB key solves the problem. Clear any text you may have in VIEW using the NEW command. Press the TAB key. The cursor 'jumps' several characters to the right. Type:

1

and press the TAB key. Again the cursor jumps several character spaces to the right. Type

11 <TAB>

and then

21 <TAB>

and finally

Move to the next line by pressing RETURN and enter the next row of figures:

<TAB> 2 <TAB> 12 <TAB> 22 <TAB> 32 <RETURN>

Build the rest of the table yourself.

If you examine the ruler with the figures you will find that the first character in each number lies under an asterisk. As we can edit the ruler we can change the position of the TAB stops simply by editing the asterisks into the correct position. Move to the next line and insert a new default ruler (CTRL-I5), and edit it to look like this. Remember to use F8 and F9 to insert and delete characters:

10 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100

Move to a new line and now enter a few rows of figures using the TAB key to separate each one. Using the first four lines from above you will end up with a table looking something like that shown in figure 3.5.

F	1	2	3	4	5
1	11	21	31		
2	12	22	32		
3	13	23	33		
4	14	24	34		
5	15	25	35		

Figure 3.5. Using TAB's to tabulate figures.

The columns of figures have again aligned themselves with the asterisks. Move the cursor back to the ruler and place it immediately after the third asterisk, below which should be the column of figures headed by 21. Now press **f9** five times to delete five full stops. The ruler becomes smaller and the asterisks move away from the column of figures starting at 31. Press **RETURN**, and the column of figures starting at 31 will re-align themselves with their host asterisk. In a similar manner, try inserting more full stops between two columns of figures (using **f8**) and press **RETURN**. Again the columns of figures automatically re-align themselves under their parent **TAB** stop.

Consider what would have happened if you had entered the four columns of figures using spaces between the columns rather than TAB stops. Initially there would have been no visual difference.

However, should you then have wished to reposition one, two, three or all four of the columns, you would have been faced with the time consuming task of deleting the space characters between the appropriate rows. By using TAB stops it is a simple and quick process to reformat numerical tables or even lists of information. So the rule is to always use TAB stops when tabulating columns - whether figures or words.

The space between TAB stops is used, but it only contains one thing, which we can't see: a single TAB character. Move to a position directly after the 1 in 21 and then press f9. The 31 in the next column closes up directly on the 21 because we have deleted the TAB character. To restore the column to its former isolation simply press the TAB key.

One final use of the TAB key is to allow you to indent the start of a new paragraph. Simply press TAB before you start the paragraph, then carry on typing. Of course you can edit your ruler and position the first TAB stop asterisk so that the size of the indent suits you.

Printing Text

Printers and the printing of documents are discussed extensively in a later chapter, but if you have a printer you may now want to print out onto paper what you have typed - this is termed the 'hard' copy.

I'll assume that your printer is connected correctly and switched on - if it isn't then refer to your printer manual for details on how to go about it. With some text already entered into Edit mode, press ESCAPE to enter Command mode. If you are using a single sheet of paper, insert this into the printer so that the printing head is near the top of the paper. If you are using continuous stationery then feed it through the printer until the printing head is just on or very slightly below a set of perforations. The command to print the current document is simply:

SHEETS

After you have pressed RETURN the prompt:

Page 1..

will appear on screen. This is informing you that VIEW is ready to send page 1 of your document to the printer. Press any key and printing will commence. As we have only a few lines in our document it will be printed on the first page. On completion of printing, the

remaining lines on the page, which will be blank, will be sent to the printer, which will then eject the sheet.

If nothing happens when you press a key after the Page 1.. prompt then check all your printer connections and ensure that the printer 'On line' light is illuminated. Again, refer to your handbook which should contain trouble-shooting details. On the other hand the printer may be happily printing away but all on the same line! This is because line feeds have not been enabled on your printer. This is easily overcome. When printing has finished simply enter the following star command in Command mode:

*FX6

and try again. This will almost certainly cure the 'problem' - just remember to do this each time you switch your printer on. A permanent solution would be to set the appropriate DIP switch inside the printer so that linefeeds are permanently enabled - again consult your printer manual for details on how to do this.

4 : Saving & Loading Text

If there is one golden rule in wordprocessing, then it is simply, 'Save your text regularly.' I can guarantee that at some stage in your wordprocessing career you will accidentally lose what is held in VIEW at the time. The fault may be yours - you switch the machine off, or accidentally catch your foot in a power cable - or it may be out of your hands - a power failure or blown fuse, for instance. But by saving regularly to tape or disc you will not lose all your valuable text. How often you save is up to you; as a guide I always save every four or five hundred words or so, or when I come to a break in the text, or if I'm off for another can of beer from the fridge.

Saving Text

How you go about saving your text will depend on what type of filing system you are using, so each of the main filing systems are discussed below. However, this is not a tutorial on how to use your filing system. For that, consult either your micro's User Guide or the accompanying DFS, NFS or ADFS Guides.

Cassette Filing System (CFS)

All you need is a suitable blank tape, clearly labelled VIEW Files, or whatever the subject of the document is. To save a file, enter Command mode and simply type:

```
SAVE <filename>
```

where <filename> is the name of the document, up to 10 characters long. After pressing RETURN, the normal tape prompt will appear:

```
RECORD and RETURN
```

Start the cassette player recording and press RETURN. When the VIEW prompt reappears, the document has been saved. I would strongly recommend that you always make a backup copy on another tape. Cassette tape storage for computer files is generally but not totally reliable, and also rather time consuming.

Disc Filing System (DFS)

You will need a suitably labelled and formatted disc - see your Disc Filing System manual or User Guide about this. Note that formatting a disc may destroy any text you have in VIEW, so make sure you always have several blank formatted discs to hand before starting your VIEW session. Insert the disc into the drive and type:

```
SAVE <filename> <RETURN>
```

where <filename> is your chosen name for the file, up to 7 characters long.

To remind yourself what a file saved on disc is about, it is a good idea to prefix each file with a directory letter (again see your manual for a description of directories). As these are VIEW files I like to save them in directory V. For a file called TEXT this would be simply:

```
SAVE V.TEXT <RETURN>
```

Advanced Disc Filing System (ADFS)

The Advanced Disc Filing System is a hierarchical filing system in that it, unlike DFS, allows directories to contain sub-directories and so on. As for DFS you will need to have a formatted disc to hand, which may or may not contain other files. Before the new disc can be used, it must have its catalogue read into memory (ie, have the names of the files on the disc read into the computer's memory). This is easily done. Insert the disc and type:

```
*MOUNT <RETURN>
```

The drive will access the disc. Typing:

```
*CAT <RETURN>
```

will list the directories and files on the disc. It is best to save all text files in a suitably named directory, typically called VIEW. This would be created with the command:

```
*CDIR VIEW <RETURN>
```

A file called Text could then be saved in the directory called VIEW with:

```
SAVE VIEW.Text <RETURN>
```

VIEW will not allow the complete filename route to exceed 19 characters.

Network Filing System (NFS)

If you are using a network then the ADFS details above will apply. Consult your network manager for details.

Loading Text

Text can be loaded simply by specifying the filename, as detailed above, and using the LOAD command instead of SAVE. A word of warning: ensure that you have saved any vital text within VIEW as the loading process will overwrite any text you may already have in memory. The following examples will load the files saved above:

```
Tape   :   LOAD <filename>
Disc   :   LOAD V.Text
ADFS   :   LOAD VIEW.Text
Net    :   LOAD VIEW.Text
```

After a LOAD operation the screen will reset itself, and you will notice that the Editing line in the status information will contain the filename of the file just loaded and the Bytes free count will have been updated. For example, if before loading any text the status information read:

```
Bytes free 12000
Editing No File
Screen mode 3
```

after loading an ADFS file called 'VIEW.Text' it would say something like:

```
Bytes free 10123
Editing VIEW.Text
Screen mode 3
```

Quick Save

To make life a little easier, VIEW provides a short form of the SAVE command. If you simply type in:

```
SAVE
```

and press RETURN, VIEW will use the filename specified in the status information. This is useful, but it can be dangerous. Once you have finished a document always clear it from memory by using NEW to

reset the **Editing No File** message. If you don't and then use **SAVE**, you could write over a previously saved file. In versions 3.0 and above of **VIEW** there is a **NAME** command which can be used to reset the **Editing** name to anything you wish. Thus the command line

```
NAME VIEW.Letter <RETURN>
```

will set the **Editing** information to:

```
Editing VIEW.Letter
```

Locking Files

If you are using **DFS**, **ADFS** or **Network**, get into the habit of locking files when you have saved final versions - this will prevent them from being accidentally overwritten. Locking files is performed via the ***ACCESS** command. To lock the files saved earlier:

```
DFS      :  *ACCESS V.Text L
```

```
ADFS     :  *ACCESS VIEW.Text LWR
```

```
NFS      :  *ACCESS VIEW.Text LWR
```

The **L** signifies the file is to be locked. Trying to save over the file with another of the same name will now result in the error message:

```
File locked
```

and the save operation will be cancelled.

Of course you may wish to re-save a modified or updated version at some later date - this is possible by unlocking the file. The procedure is the same except that the **L** is omitted from the command thus:

```
DFS      :  *ACCESS V.Text
```

```
ADFS     :  *ACCESS VIEW.Text
```

```
NFS      :  *ACCESS VIEW.Text
```

Once you have re-saved remember to re-lock the file!

Filename Choice

The choice of filename is an important one - it must reflect the contents of the document, so that when you are looking for it, you will know what to look for. File names such as **LET1** and **LET2**, or even **LETTER1** and **LETTER2**, are meaningless a day or two later, let alone a week.

When using DFS the filename choice becomes more important still, as we are limited to just seven characters. Organisation here is vital. Ideally, all similar text should go onto a disc labelled for just that purpose. This book was written using VIEW and a disc was dedicated to it. The disc was labelled:

VIEW Book

Each chapter was then saved using its number, ie:

Intro Chap1 Chap2 Chap3

and so on. No problems here. If you write a lot of memos and letters, dedicate a disc directory to the type of letter or memo and list these on the disc label. Memos to the Managing Director would be placed in directory M, letters to clients placed in directory C and so forth. The best filename to use is generally the date of the letter in numeric form. Use the American way of specifying the date, ie, month first. So a letter typed to a client on 1st December 1986 would be saved as follows:

SAVE C.120186

which breaks down as follows:

C = directory C is Clients 12 = December 01 = 1st 86 = year

A filename of M.092386 would indicate that it is a memo sent to your MD on 23rd September 1986. The reason for using the month first in the filename is one of convenience. All files will fall into 12 categories, 01 to 12, rather than 31 if the date was used first. It also makes it much easier to locate a particular memo as you can, when looking down the disc catalogue, go straight to the month and then select the date.

If you are writing many such files then it would be much better to dedicate a disc to each, suitably labelled: VIEW MEMOS, VIEW LETTERS and so on. There are still limitations though, particularly when using DFS, as this normally limits you to just 31 files per disc. This is where ADFS with its sub-directory structure comes into its own.

The above techniques apply when using ADFS and NFS. However ADFS allows us to be even more specific. Suppose, for example, we are writing a monthly report, say for May, for the local hockey league, we could create a directory called Hockey on our disc labelled VIEW. Proceed as follows:


```
*DIR <RETURN>
*CDIR Hockey <RETURN>
```

Now we could save the May report as follows:

```
SAVE Hockey.May <RETURN>
```

Of course I could have gone further and created a Report directory within the Hockey directory to be even more specific. Creating the Report directory can be done from within the Hockey directory as follows:

```
*DIR <RETURN>
*DIR Hockey <RETURN>
*CDIR Report <RETURN>
*DIR <RETURN>
```

Thus a filename such as:

```
SAVE Hockey.Report.May <RETURN>
```

would leave us in very little doubt as to the contents of the file. If there were several reports for May then May could be made into a directory and the date method for saving the file could then be used.

Using ADFS, we are not limited to a set number of files, though we are of course limited by the physical capacity of the disc itself. If you anticipate the need to save many small documents then it is much better to use the ADFS filing system. This is fitted as standard on Master 128 and Master Compact but must be fitted to standard BBC B and B+ micros by way of an upgrade. This can be done by your local Acorn-approved dealer.

Disc Commands

When you type characters into VIEW, they are stored in your computer's memory. If you are using DFS, ADFS or a Network then there are certain commands that you must be very careful not to use, as they also require the use of the computer's memory and will almost certainly overwrite what is already in memory, erasing your document irretrievably. Table 4.1 lists 'safe' and 'unsafe' commands. The safe commands can be used at any time without fear of corrupting text. The unsafe commands should only be used after your text has been safely stored on your filing system medium for subsequent re-loading.

Safe Commands: *ACCESS, *APPEND, *BACK, *BUILD, *CAT,
 *CDIR, *CLOSE, *CREATE, *DELETE,

*DESTROY, *DIR, *DISMOUNT, *DRIVE, *DUMP,
*EX, *FREE, *INFO, *LCAT, *LEX, *LIB,
*LIST, *MAP, *MOUNT, *OPT, *REMOVE,
*RENAME, *SAVE, *SPOOL, *SRLOAD, *SRSAVE,
*TITLE, *VERIFY.

NB. Commands *SRLOAD and *SRSAVE are only safe if the Q option is omitted.

Unsafe Commands: *BACKUP, *EXEC, *COMPACT, *COPY, *FORM
*FORMAT, *LOAD, *PRINT, *RUN, *SRLOAD using
Q, *SRSAVE using Q, *TYPE.

Table 4.1. Safe and Unsafe Filing System Commands.

VIEWing Saved Files

When you have saved a document to disc (for the rest of this book the term 'disc' will be used to encompass all filing systems) it is possible to examine it without disturbing text already in memory. This is done using the SCREEN command which was introduced earlier. To preview a file called V.TEXT, the command would be:

```
SCREEN V.TEXT <RETURN>
```

Obviously, the disc containing the file must be to hand and on the currently selected drive. Drive numbers may be used as part of the file handle. For a file called V.MEMO on drive 2 the command:

```
SCREEN :2.V.MEMO <RETURN>
```

would preview the file. Pressing the ESCAPE key at any time will abort the preview.

Bootiful

A most useful feature of disc based systems (DFS, ADFS and NFS) is their ability to use !Boot files. These are files which contain a series of commands that can be executed automatically by the computer to set or configure the way in which you wish to use the machine. For example, each time you turn the computer on to start a VIEW session, you may want first to select VIEW and then set the screen mode. In commands this would be:

```
*WORD <RETURN>
MODE 3 <RETURN>
```

We can include these two commands in a !Boot file on the disc, so that when we turn on the micro and insert the disc they will execute almost automatically. To write the !Boot file, enter VIEW, type NEW, go into Edit mode and enter the two commands one after the other as shown above, pressing the RETURN key after each. ESCAPE back into Command mode and then save the text using the WRITE command. The !Boot file must be in the \$ directory or your network root directory to operate correctly. This is done as follows:

```
DFS   :   *DIR $
ADFS  :   *DIR
NFS   :   *DIR
```

Now the !Boot file can be written to disc:

```
WRITE !Boot <RETURN>
```

Finally type:

```
*OPT 4,3 <RETURN>
```

The disc will whirr briefly. This last command is writing some information to the disc about the way in which the !Boot file is to be treated. Remove your disc from the drive and switch your micro off for a few seconds. Turn it back on and insert the disc back in the drive. Now the disc must be booted. This is done by holding down the SHIFT key and pressing the BREAK key before releasing both. Watch carefully and you will see that both commands are executed to boot your system ready to use. As Compact users have VIEW supplied on disc it will be necessary to load this into memory first in the normal manner.

The !Boot file can be made quite complex to undertake a whole variety of tasks. We will be adding to it as we progress.

5 : Markers

VIEW allows sections of text to be identified by enclosing it within bounded areas. These areas are defined by the use of markers, and VIEW has a set of 6 markers. The text to be marked may vary from just a few characters up to a few hundred lines. The VIEW markers are referred to by their number, and the first two markers, numbers 1 and 2, are used most frequently as these are the two that can be seen on screen. In modes other than Mode 7, they appear as an inverse character - black on a white background instead of the normal white on black. In Mode 7 they appear as an asterisk and minus sign. The remaining markers, numbers 3 to 6, are not visible on the screen but can be found when needed.

Markers have many uses: to allow you to find your way quickly to a certain point in a large document; to move a section of text from one part in a document to another; to copy a repetitive section of text quickly and simply; and to allow you to delete a section of text. So get used to the markers: they play a very big role in effective wordprocessing.

Moving Text

The ability to move sections of text around within a document is one of the main features of a wordprocessor such as VIEW. For instance, you might wish to move a paragraph from the start of a letter to its very end. Using a conventional typewriter you would have to re-type the whole letter. With VIEW you simply place markers at the start and end of the section of text, move the cursor to the point in the document where you wish the marked section of text to be moved to, then press the MOVE BLOCK function key, SHIFT-f0. The block of text is moved instantly.

Inserting a marker is straightforward. Move the cursor to the desired point. Once there, press the SET MARKER function key, SHIFT-f7. At this point the inverse letters MK appear in the top left hand corner of the ruler. We must now inform VIEW which marker we wish to insert at this point - this is done by pressing the marker's number on

the keyboard, ie, one of the numbers 1 through to 6. The MK letters disappear from the ruler and the marker will have been inserted.

Let's have a go. First some text is needed to work on. Anything will do, but you might like to enter the snippet shown in figure 5.1, using the ruler as shown to get the correct format. Many of the examples in this book will use this text to illustrate ideas, so after entering it you might like to save it for later recall. Use the filename LUNAR.

The Times Sunday July 21 1969

Nell Armstrong became the first son to take a walk on the moon's surface today. The spectacular ascent came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Figure 5.1. The LUNAR sample text.

We can try moving the entire first paragraph so that it comes after the second paragraph. Enter Edit mode and move the cursor to the start of the first sentence, the 'N' of 'Neil'. Then insert marker 1 in the text as described above (press SHIFT-17 followed by 1). A white block should appear on the screen with the letter N inverted inside it.

Insert marker 2 next - move the cursor to the end of the second sentence, which is also the end of the first paragraph (the last word being 'craft.') and press SHIFT-7 and then the 2 key to insert marker 2. To move this marked block of text we must first move the cursor to its destination - in this case on a new blank line after the last line (which is "Eagle has landed."). Once here, the move is effected simply by pressing MOVE BLOCK, which is SHIFT-10. The block will instantly be moved to its new position, leaving you with a slightly rearranged news story as shown in figure 5.2.

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Nell Armstrong became the first woman to take a walk on the moon's surface today. The spectacular moment came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

The procedure for copying text is very similar to that for moving blocks, but it has a different result: the original marked text is not deleted after it has been copied elsewhere. This facility is most useful if you wish to insert a repetitive piece of text at several points in a long document. We could add a title to our current document by copying a few words from within the text to the top. For example, the words at the end of what is now the first paragraph,

sition the markers by moving the cursor to the correct position and pressing SHIFT-17 1. This requires the markers to be placed under the 'e' (to exclude the quotes) and under the full stop after 'base' (to include the full stop). Marker 2 can be placed after the 'e' in 'base', over the full stop, with SHIFT-17 2. As before you simply need to move the cursor to the position you want the text copied to. Move the cursor to the top of the first paragraph placing it under the 'T' in 'The'. Before we can copy the text we need to make a space for it - a blank line. This is done by pressing function key f6, INSERT LINE. Be careful: it is directly next to f7 which has the opposite effect and will DELETE LINE. The copying process is done by pressing the COPY key at the lower right of the keyboard. Figure 5.3 shows the result.

The Times Sunday July 21 1989

Tranquility base
The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from earth on the moon came from Aldrin: "Tranquility base. The Eagle has landed".
Neil Armstrong became the first man to take a walk on the moon's surface today. The spectacular moment came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

Figure 5.3. Copying text to provide a title.

As already mentioned, unlike deleting text, a text copy operation does not remove the markers. This is quite deliberate as it allows us to copy the same section of text once again, or as many times as we like. However we will eventually need to remove the two markers, so how is it done? The obvious way is to move the cursor over each marker in turn and press **F9**, **DELETE CHARACTER**. This has a side effect in that it will also delete any character over which the marker was sitting. The best way is to **ESCAPE** into command mode. You will probably notice straight away that the status information contains an extra line:

Marker(s) set 1,2

Type the following command:

CLEAR

The marker status information will be deleted. If you ESCAPE back into Edit mode you will find that the markers have been removed from the text.

The only other refinement we may wish to make to the above text is to change the 'b' in 'base' into a 'B'. This can be done in one of two ways. The obvious one is to move the cursor so that it sits under the 'b' and overtype it with a capital 'B'. This is fine for single letter changes. An alternative method is to move the cursor under the b and then use function key SHIFT-F1 SWAP CASE, which will alter the case of the character under which the cursor sits. This method comes into its own if you wish to invert the case of several letters or perhaps a whole sentence. Simply hold the SHIFT-F1 keys down and it will continue to invert and move to the next character until you release both keys. You might like to re-save the LUNAR text at this stage.

Deleting Text

There are four ways in which text, other than single characters, can be deleted in VIEW. Each method has a specific action: to delete from the current cursor position to the end of a line; to delete an entire line; to delete up to a specific character; and finally to delete a block of text. It is most important to remember that these actions are irreversible, so make sure you really do wish to make the deletions planned. In fact it's best always to save your text to a temporary file before you make any large scale deletions - just in case!

To delete to the end of a line we first need to move the cursor to the point within the text from which we wish to delete. The deletion will start with the character immediately above the cursor. Say we wish to delete the last four words in the first line of our LUNAR text, the 'was near perfect and' bit. First move the cursor so that it sits under the 'w' of 'was' and then press function key F3 DELETE END OF LINE. Instantly the four words are removed to leave the text as shown in figure 5.4.

The Times Sunday July 21 1969

Tranquility Base
The landing in the Sea of Tranquility,
the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".
Neil Armstrong became the first man to take a walk on the moon's surface today. The spectacular moment came after he had inched his way down the ladder of the fragile lunar bug.
Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

Figure 5.4. Deleting to the end of a line.

Deleting a whole line is even more straightforward. You just put the cursor anywhere on the line you wish to delete and press function key **F7 DELETE LINE**. Try deleting the very last line in the text - position the cursor anywhere on the last line and press **F7**.

Occasionally you will want to delete just part of a line, perhaps a few words, and this is best done using function key SHIFT-F3, DELETE UP TO CHARACTER. The technique is easy. First move the cursor to the first character which you wish to delete and press SHIFT-F3. An inverse CH will appear in the top left of the ruler - VIEW is waiting

for you to enter the character which you wish to delete up to. Press the appropriate key, noting that case is significant - VIEW differentiates between a lower case d and an upper case D. In the LUNAR text we could delete the text 'The spectacular moment came after' from the second line of the second paragraph. First move the cursor to the 'T' in 'The' then press:

SHIFT-f3 r

However, in this case there is an 'r' at the end of 'spectacular' and so only text up to here will be deleted - it is necessary to perform the command once again to complete the deletion. One way around this would be to put a special marker character at the end of the text to be deleted. For example the @ character could have been entered after the 'r' in 'after', and then entering:

SHIFT-f3 @

would delete the entire section. If two or more characters of the type to be deleted are side by side then all of them will be deleted. Placing the cursor under the 'T' in the following text:

He called for you several times.

and then entering:

SHIFT-f3 l

will delete both 'T's in 'called', leaving:

ed for you several times.

The SHIFT-f3 function will not work across several lines: its action is limited to the current line only.

Deleting a section of text is performed first by marking the text in the manner already described and then pressing CTRL-f0 DELETE BLOCK. Experiment on some text for yourself - if you have entered the LUNAR text try deleting the last paragraph.

A second word of warning - I make no apologies for repeating myself - always double check your actions, especially when using function key f0. Pressing CTRL in conjunction with f0 when you really meant to press SHIFT could prove disastrous. Again, always save your text regularly and always before any sort of action on a block of text.

Markers 3 to 6

Markers 1 and 2 must be used for general block operations, a few more of which are discussed below. Markers 3,4,5 and 6 are very useful to act as reference points. Imagine we are working on a rather long document which requires a form of cross referencing best done by moving between the two sections alternately. The long winded way to perform this would be to use the cursor keys to move up and down through the text. A much better way is put a marker at the start of each section: more specifically one of markers 3 to 6. Again, let's try an example. In Command mode type NEW to clear any text and then type

1. This is section 1 of the text

Next press the RETURN key about twenty times to move you some way down the screen and enter:

2. This is section 2 of the text

Again press the RETURN key about twenty times and type:

3. This is section 3 of the text

As things stand, it is quite easy to move between sections 1 and 3 of the text simply by using function keys f1 and f2 (TOP OF TEXT and BOTTOM OF TEXT). However, section 2 and any future sections that may be inserted between sections 1 and 3 are a different matter. Well, no they aren't. Move to the 2 in section 2 using the cursor keys, and once here set marker 3, by typing:

SHIFT-f7 3

Remember, markers 3 to 6 are invisible on screen, so nothing will appear apart from the MK in the ruler after pressing SHIFT-f7. But ESCAPE to Command mode and the status information will state:

Marker(s) set 3

Press f1 to move to the top of the text. We can get straight back to marker 3 by using function key SHIFT-f6 GOTO MARKER. To do this press SHIFT-f6 and then the number of the marker, 3 in this case. You are instantly taken to section 2, the home of marker 3. Of course you can use markers 1,2,4,5 and 6 in a similar way: press the relevant number key after SHIFT-f7. Deleting all markers from your text is easiest done in Command mode using the CLEAR command. If you wish to delete individual markers you must first move to the marker concerned, using SHIFT-f6, then f9 DELETE CHARACTER. One point to

note - if your marker is set over a character then deleting it in this manner will also cause the character to be deleted, so it will need to be re-typed.

Saving Marked Text

Markers can be used to mark out a section of text of any length within a document, which can then be saved independently of the rest of the text. Again, there's nothing complicated about the procedure. First insert the markers at the start and end of the section you wish to save. ESCAPE to the Command mode and enter:

```
WRITE <filename> <marker no.> <marker no.>
```

The command WRITE acts similarly to SAVE, which would save the entire document. The parameter <filename> is the name of the file which the command will WRITE to. Finally the <marker no.> parameters are the marker numbers. Therefore if we had some text surrounded by markers 5 and 6 we could write this to a file called TEMP using:

```
WRITE TEMP 5 6
```

It makes good sense to use markers 3 to 6 for this sort of thing, leaving markers 1 and 2 free for other operations. If you specify a marker that has not been set the error message:

```
Marker not set
```

will be displayed.

Once you have saved your marked section of text it can be used however you wish at a later stage, just as though you had originally saved using the SAVE command.

Loading text to a marker

A marker can also be used to load a section of text to anywhere within or to the end of a document. The LOAD command cannot be used for this purpose as it will wipe out existing text - the necessary command is READ, which takes the following format:

```
READ <filename> <marker no.>
```

First insert a marker at the point where you wish the new section of text to be inserted. If marker 3 is set and the file to be read in is called TEMP the command, in Command mode, would be:

READ TEMP 3

The file will then be read in and deposited in front of the marker, so that when the operation is finished, the marker will be at the end of the newly read section of text.

If the text to be read in is longer than the space remaining within VIEW (ie, the number of characters in the file exceeds the Bytes free value), then only text up until that point will be read in. In VIEW 3.0 and subsequent versions the message:

Not all read in

will be displayed. In VIEW 2.1a and 1.4 no such message is provided, so it is best to check if you are reading in a long file. Of course, more bytes can be freed simply by going to a lower resolution screen mode. For example if you are operating in Mode 3, enter Mode7 and then read the file in. Once read in type Mode 3 to restore the original mode. If the file is to long then the message:

Not enough memory

will be displayed. In this case you have three options: continue in the current mode; delete enough text to allow you to go into Mode 3 or the best solution - split your file into two smaller ones by WRITEing the first half to a new file.

Counting within Markers

Many of VIEW's commands can have their action limited within boundaries of two markers. A command already familiar is COUNT, which counts the words. By inserting two markers around a block of text, the number of words within the block can be counted using the syntax:

COUNT <marker no.> <marker no.>

Thus if some text was bounded by markers 1 and 2, the number of words within it could be obtained with the command:

COUNT 1 2

The COUNT command entered without <marker no.> parameters still works on the whole text. Many other VIEW commands can have their action limited to marker boundaries - full details of how to do those will be given when we look at the commands in question.

6 : Formatting & Justification

We have already touched briefly on the subject of formatting text. To recap: with formatting enabled, when your typing reaches the end of the ruler the text will wrap around onto the next line, or, to be more specific, if the word you are currently typing will not fit within the bounds of the ruler the word will be moved down to the start of the next line.

The top right-hand corner of the ruler contains information on how VIEW will arrange the text being typed. There are three indicators: an F shows that formatting is enabled, an I indicates that VIEW is operating in its Insert mode and a J indicates that VIEW will justify your text. If F, J or I are missing then the respective option is disabled. We have described formatting in Chapter 3, so let's now tackle the other two in turn.

Insert Mode

Up until now, when we have decided to edit a word, the cursor has been moved to the relevant point in the word and **f8** pressed to insert a character. Alternatively, we can type and overwrite the letters which make up the word. With Insert mode enabled, any characters typed at the keyboard will be inserted at the point of the cursor. In effect, the **f8** key is always pressed for you. To enable Insert mode you must press function key **CTRL-14**, **INSERT/OVERTYPE**. Pressing **CTRL-14** again will disable Insert mode - remember you can tell whether you are in Insert or Overtyping mode by looking at the ruler. To see the difference between these two modes, clear any text you may have in VIEW, enter Edit mode and ensure that Insert mode is disabled, ie, that no I sits in the right-hand of the ruler. Now enter this text:

On Olympus Top Most Top A Fat Greasy Vulture Suns
Himself

Now, the second word should read 'Olympus' - we have missed out an 'm', typed 'o' instead of 'u' and added an extra 's'. To remedy this, move the cursor so that it is under the 'p' and then type four letters, 'm', 'p', 'u' and 's' to leave:

```
On Olympus Top Most Top A Fat Greasy Vulture Suns  
Himself
```

The 'poss' of 'Olyposs' has been overtyped with 'mpus' to give 'Olympus'. Those who know some human biology might recognise that little sentence. It's a mnemonic, providing the initial letters of all the 12 cranial nerves. Well, nearly all: we need to add the word 'Old' between 'On' and 'Olympus'. To do this, enable Insert mode by pressing function key CTRL-*I*, and note that an *I* is now present in the ruler. Move the cursor so that it sits under the 'O' in 'Olympus' and type 'Old'. As you type, the rest of the line to the right of the cursor will move along to the right a character at a time and the letter typed will be inserted. After typing the 'd' in 'Old', press the space bar to insert a space and you will be left with the full saying:

```
On Old Olympus Top Most Top A Fat Greasy Vulture  
Suns Himself
```

To insert or not? It is really up to you. For safety's sake, I always have Insert mode enabled. It is easier to delete redundant text than to re-type it should you have accidentally over-written it. Another useful feature of Insert mode is that you can use the DELETE key to erase text from right to left. As you delete a letter the gap it leaves is closed up from the right.

Justification

When entering text into VIEW it is 'ranged left' - the first letters of each line are aligned one under the other. In the previous examples with formatting enabled, the text has always had a ragged right edge, ie, an uneven one. Look carefully at the letter at the very start of Chapter 1. The text is not ragged on the right-hand side; just like the left-hand side the words, or more specifically the last letters in each word, all line up and so we have a neat, non-ragged page of text. This is justification, and the text is said to be justified.

Therefore with justification and formatting enabled your text will have both edges justified. VIEW justifies by padding out short lines, inserting extra spaces between words.

Justification is turned on and off by the function key CTRL-*J*, JUSTIFICATION. The presence or absence of a *J* in the left-hand side of the ruler will tell you what the current state is. To see how it works, first ensure that Justify mode is disabled by pressing CTRL-*J* until no *J* is present in the ruler. Also, enable Format mode (CTRL-*F*). Enter several lines of text: at least four or five are needed - copy this

paragraph if you are stuck for ideas. Keep a copy of this text on disc or tape to load back in in a few moments time. Now move the cursor to the top of the text and enable justification by pressing CTRL-J. Nothing happens. Indeed nothing will, until you tell VIEW to act by pressing function key F0 FORMAT. After this, the text will be justified almost immediately. Move the cursor to the top of the text once more and disable justification. Now press F0 - the text returns to its non-justified state. Figure 6.1 shows the sample LUNAR text before and after justification.

The Times Sunday July 21 1969

Tranquility Basin

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Nell Armstrong became the first man to take a walk on the moon's surface today. The spectacular moment came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

[illegible]

The Times Sunday July 21 1969

Tranquillity House

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Neil Armstrong became the first man to take a walk on the moon's surface today. The spectacular ascent came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

중립적 입장에서 남북한을 비교한 연구는 1990년대 이후에 나타난다. 이 시기에는 남북한을 비교한 연구가 대량으로 발표되면서, 남북한을 비교한 연구의 양과 질이 크게 향상되었다. 이 시기에는 남북한을 비교한 연구의 양과 질이 크게 향상되었다. 이 시기에는 남북한을 비교한 연구의 양과 질이 크게 향상되었다.

Figure 6.1. The LUNAR text before (top) and after (bottom) justification.

Clear the text from VIEW and then reload the previously saved text. Move to the bottom and press the RETURN key a couple of times to insert blank lines. Now read in some text to the current cursor position by entering Command mode and typing:

READ Temp

where Temp is the filename, to give you two copies of unjustified text. Save this over the previous file. Now move the cursor to the top of the first paragraph, and with justification enabled press f0 to FORMAT. Note that only the first paragraph of the text is formatted; the second is not touched. This is useful as it allows us to include formatted and justified text together with unjustified text (tables and figures perhaps). To format the next paragraph in the same manner, the cursor must be moved to its start and then f0 pressed (in fact, simply pressing f0 a couple of times will move you to the start of the next paragraph).

There are times, however, when we will wish to format and justify a whole document and clearly this method is a little labour intensive. Load in the resaved text and then make a carbon copy (by using markers and the COPY key) of the two paragraphs to make four. Set yourself a new default ruler and edit it to a smaller size than the current one, then ensure justification is enabled. Now return to Command mode and enter:

FORMAT

This command will then cause the whole of your text to be formatted, regardless of the cursor position in the text. As the process can take several seconds for long documents, dots are printed on the screen as each paragraph is formatted. If you return to Edit mode you will see that your text has been fully justified. To revert it to its non-justified state, disable justification and in Command mode simply enter the command FORMAT once gain.

In long documents we may not wish to format the entire text but merely a substantial portion. We can limit the range of the FORMAT command by using markers 1 and 2. Simply set the desired limits of the format by inserting markers 1 and 2 at the start and end, then use the FORMAT command as follows:

FORMAT 1 2

Only the text within their bounds will then be formatted.

By Command

While the Format, Justify and Insert modes can be set using function keys, it is often much quicker to use a command provided to set their state - this command is SETUP and is available from version 3.0 of

VIEW. It is of course used in Command mode and if followed by any combination of the letters F, J or I will enable the relevant options. Omitting a letter will disable the option. Thus the command:

SETUP F.11

will enable Insert mode and turn formatting and justification on. The command

SETUP FI

will enable Format and Insert modes, but turn justification off. All three can be disabled simply by typing:

SETUP

The advantage of the SETUP command is that it can be included in a !Boot file on disc. If you saved the previous !Boot file, load it into VIEW and simply insert a new line at the end, tailored to your own needs - it might now look like this:

```
*WORD
MODE 3
SETUP F
```

and then save this back to disc with the WRITE command, ie.

WRITE IBoot

More on formatting

Before formatting a paragraph, look at it closely to ensure the result will be what you want. Formatting text can sometimes destroy the layout you have designed, rendering it useless. As such it is important to understand what VIEW regards as a paragraph, or more importantly how VIEW recognises the end of one. Work through the following example to see just how formatting works. Clear any existing text from VIEW (using NEW) and SETUP FJ1. Now edit a new ruler to delete characters up to the third tab stop and enter the text exactly as shown in Figure 6.2

```
F *.....*.....*.....*.....*.....*.4  
..*.....*.....*.....*.....*.....*.4  
I intend to bring the following items:  
1) New microcomputer  
2) New disc drive  
I hope this will be okay with you.  
*****
```

Figure 6.2. Some sample text prior to formatting.

Now move the cursor to the top of the text and press function key f0. The result is that the four lines of text are all joined together or 'concatenated', thereby destroying the layout, as shown in figure 6.3.

```
F | .....*.....*.....*.....*.....*.....*.....*.....*.....*.....C
..| .....*.....*.....*.....*.....*.....*.....*.....*.....*.....C
   | Intend To bring the following items: 1) New
   | microcomputer 2) New disc-drive I hope this will
   | be okay with you.
```

Figure 6.3. After formatting the layout has been destroyed.

VIEW formats the entire paragraph, so to prevent this sort of thing it is important to know exactly what VIEW regards as the end of a paragraph. These are the ends it recognises:

- * A blank line
- * A line that begins with a space
- * A line that begins with a TAB
- * A new ruler
- * The end of the document
- * A line that contains a stored command
- * A line with text in the left margin

The last two may not mean much at this stage as we have yet to discuss margins or stored commands. To avoid lines being concatenated as above, we could have inserted a space at the start of each line and then formatted the text. With the cursor on the first line, pressing function key f0 will cause only the first line to be formatted and the cursor will move down to the second line. Note that although justification is enabled the line does not get ranged left and right - VIEW has a degree of intelligence in such cases and has realised that to do this would leave unacceptable gaps in between words, so has left just a single space between each word. There is not much to be gained by formatting the second line. The fourth line does not have a space inserted before it, so formatting line 3 would result in line 4 being tagged onto its end - so avoid formatting line 3 or insert a blank line between them by putting the cursor on line 4 and pressing function key f6 INSERT LINE - be careful as f7 is DELETE LINE!

It is especially important to bear in mind what VIEW regards as the end of a paragraph when you are performing a total document format using the FORMAT command. Once this has started you have no control on what is and what isn't formatted. If in doubt always insert a blank line at the point which worries you - this can be deleted afterwards - or save your text before formatting.

7 : Search & Replace

Perhaps the most well known feature of wordprocessors is their ability to go through documents and change, delete or replace selected words or phrases each time they appear in the text. If you were mailing a list of people about a product or meeting, sending a few letters to friends or distributing a circular, you could personalise each one simply by substituting the name each time. A letter might start off as follows:

Dear Marc

and then use Marc's name within the text several times. Once you have printed the letter, you can replace every occurrence of Marc with a new name, for example Sarah, with the minimum of fuss - the sort of task outlined at the start of Chapter 1.

If you were preparing a document for both British and American markets, subtle variations in spelling could be a big irritation. But use of certain VIEW commands would allow you to replace all occurrences of the word 'colour' in the British document with 'color' for the American version.

Facilities for performing these tasks and variations on them are available for use in VIEW by three Command mode commands: CHANGE, SEARCH and REPLACE.

A Folding Case

Before examining these commands in detail, let's look at another one. First a question: what is the difference in meaning between the following words:

Tranquillity TRANQUILITY TranQuillity
tranquillity

The answer is nothing! They all convey the same information, but to VIEW they can appear to be all the same or four different words. Obviously, the key lies in the case of the letters, whether they are capital (upper case) or small (lower case) letters. The four words in the example use either the same case letters or a mixture of both.

The manner in which VIEW recognises and treats the case of letters is controlled by the command FOLD. ESCAPE into Command mode and enter the command FOLD. After pressing RETURN, a message will be displayed. It is likely to be:

Folding on

but it may be:

Folding off

With folding ON then cases of the same letters will not make any difference. The words TRANQUILITY, Tranquility and any other combination of upper and lower case letters will all appear the same. With folding OFF then the case of the characters is taken into account, so that in this example the four versions of tranquility would be recognised as different words by VIEW.

Folding is turned on and off by following the command with a 1 or 0 respectively, ie.

FOLD 1 - Folding on FOLD 0 - Folding off

In later versions of VIEW the commands ON and OFF may be used in place of 1 and 0, thus:

FOLD ON FOLD OFF

Remember that the current folding status can be displayed at any time by entering the command FOLD in Command mode. The way in which FOLD effects the CHANGE, SEARCH and REPLACE commands will be discussed with examples below as we now look at each of these commands in more detail.

CHANGE

The CHANGE command works globally, that is it will have an effect on all occurrences of the specified word anywhere in the document. The command takes the form:

CHANGE <first> <second>

All occurrences of the <first> word will be changed to the <second> word, subject to the setting of FOLD. The following examples show how the CHANGE command works with folding ON and then OFF. The word 'Computer' is assumed to have been entered into VIEW four times using different combinations of upper and lower case characters. The CHANGE string specified is:

VIEW : A Dabhand Guide

CHANGE Computer Micro

The results are:

With Folding ON

Before CHANGE:

Computer	COMPUTER	Computer	computer
----------	----------	----------	----------

After CHANGE:

Micro	MICRO	Micro	micro
-------	-------	-------	-------

With Folding OFF

Before CHANGE:

Computer	COMPUTER	Computer	computer
----------	----------	----------	----------

After CHANGE:

Micro	COMPUTER	Computer	computer
-------	----------	----------	----------

With folding ON, the case of the word to be changed is generally preserved. If a word begins with an upper case character then it is preserved regardless of the case of the first character in the <second> word. If the <first> word is composed totally of upper case characters then the <second> word will be inserted using upper case characters. With folding OFF, only words which match <first> exactly will be changed to the exact word specified in <second>.

CHANGE will normally work globally on an entire document, but by marking a section of text with markers 1 and 2 it is possible to limit the effect of a CHANGE to a specific block of text. Thus the command:

CHANGE Computer Micro 1 2

will change all occurrences of Computer to Micro that occur between markers 1 and 2.

It is also possible to replace a sequence of words to another sequence of words by enclosing them within slashes like this:

CHANGE/<first sequence>/<second sequence>/

The slashes are known as delimiters. When VIEW encounters the slash it knows that a phrase or sequence of words follows and ends with the next slash. The phrase following the slash and up to the next slash is the replacement sequence. It is important to remember

that the first slash must come directly after the CHANGE command with no extra spaces being inserted between subsequent slashes.

Markers may be used to limit the effect of CHANGE but these must follow directly after the last slash, ie,

```
CHANGE/<first sequence>/<second sequence>/12
```

otherwise a 'Bad marker' error will be given and the CHANGE will not take place.

REPLACE

The REPLACE command acts in a similar manner to CHANGE. However, REPLACE is a selective function whereas CHANGE is global. The syntax of the command is:

```
REPLACE <first> <second>
```

When the REPLACE command is used, VIEW searches for the first occurrence of <first>, bearing in mind the setting of FOLD. On locating <first>, the Edit mode screen is entered and the cursor sits under the first letter of <first>, waiting for you to press one of the Y or N keys. Pressing Y causes VIEW to replace <first> with <second>; if the N key is pressed then <first> will be left as it stands. In both instances, VIEW will then locate the next occurrence of <first> and repeat the process until the end of the text is reached, when you will be returned to Command mode. The REPLACE command can be aborted at any time by pressing the ESCAPE key.

Why should we need this selective replace facility? Well, suppose we have a document that contains the following text:

```
The micro will beep when the micro switch is
activated.
```

Now we decide to change 'micro' into 'computer', with the desired result being:

The computer will beep when the micro switch is activated.

Using the CHANGE command we would enter:

```
CHANGE micro computer
```

but the result would be:

```
The computer will beep when the computer switch
is activated.
```

Both occurrences of 'micro' have been replaced - this is the effect of a global change. In this case we need to use the REPLACE command as follows:

```
REPLACE micro computer
```

and respond Y to replace the first 'micro' and N to leave the second unchanged.

As with CHANGE, markers 1 and 2 can be used to limit the REPLACE to a block of previously marked text. The syntax is the same:

```
REPLACE <first> <second> 12
```

The FOLD setting effects text in exactly the same manner as for the CHANGE command.

SEARCH

The SEARCH command does not change your text in any way; it simply moves the cursor to the first occurrence of the specified search word. For example, if VIEW contains the following text:

The man walked into the room. The other man watched closely.
and we used SEARCH as follows:

```
SEARCH The
```

then the Edit mode screen will be entered and the cursor will sit under the first 'The' in the sentence (ie, the first word). We can now edit the text as normal. However, at any time we can locate the next occurrence specified in the search string by pressing function key CTRL H, NEXT MATCH. VIEW 'remembers' the position of the last occurrence of the search word, so that even if you have taken the cursor past that point it will be moved back to the next occurrence.

Wildcard Search

SEARCH allows you to locate words that you may have misspelt by using a wild card to replace unknown letters. The wild card character is a 'hat' and question mark combination, ie, ^? and this should be used for each unknown letter. The command:

```
SEARCH Epidi^?^?^?^?
```

will locate occurrences of 9 letter words beginning with Epidi - no matter what the last four letters are. In practice it is not really necessary in this case to use wild cards as the command:

SEARCH Epidi

will work just as well. The wild card facility comes into its own when used in this fashion:

SEARCH A^?c^?s^?^?n

Here, VIEW will locate words whose first, third, fifth and eighth letters are A,c,s and n respectively. The wild card option is also available for use with the REPLACE and CHANGE commands, but in the latter case it should be used with extreme care, otherwise you may find all sorts of words being changed incorrectly!

In versions of VIEW prior to 3.0 the wild card character is simply a question mark. Thus to locate the four letter word whose first and last letters are A and s we could use:

SEARCH A??s

As with CHANGE and REPLACE markers 1 and 2 can be used to limit the extent of a SEARCH.

Character Specialities

A variety of special characters can be included as part of the word to be CHANGED, SEARCHED or REPLACED. Table 7.1 shows the full list for VIEW 3:

Key	Command
^C	Carriage RETURN
^L	Left margin TAB
^S	Space
^T	TAB
^Z	Hard space
^_	Highlight 1
^*	Highlight 2
^?	Wild card
^^	^ character

Table 7.1. 'Hat' search characters.

As with the wild card character each must be prefixed with the hat character. Highlight 1 and Highlight 2 will be explained in Chapter 11. The hard space character is a special space character that VIEW inserts into a line of text in order to justify it within the ruler bounds.

Using the special characters is easy. Two commonly ones are the RETURN and TAB characters, represented by the ^C and ^T characters respectively. So if we wished to search for 'time' on all occasions that it is followed by a RETURN, the syntax would be:

```
SEARCH time^C
```

Similarly, the command:

```
SEARCH ^TNumber
```

will search for occurrences of the word 'Number' where it follows a TAB character. The command:

```
SEARCH ^TNumber^C
```

would look for a TAB character followed by 'Number' and then a RETURN.

In versions of VIEW prior to 3.0, only the RETURN and TAB characters can be searched for. The syntax for these was as follows:

```
|          RETURN
~          TAB
```

Thus to search for the word 'time' at the end of a line we might use:

```
SEARCH time|
```

8 : Stored Commands

So far the commands we have encountered have all been entered in Command mode and their effect has been almost immediate. Stored commands take the form of two letters which are entered into VIEW in Edit mode, and their effect is only seen when the VIEW document is either previewed with SCREEN or a hard copy printed out.

We examined the Edit mode screen in Chapter 2, and saw how text typed in sits not to the far left of the screen but a few character spaces off to the right. It is into this margin that stored commands are entered. If you look at the top left-hand corner of the ruler the letters F and J may or may not be present. These are in effect stored commands.

A stored command is entered by moving the cursor into the stored command margin, typing the two letters representing the particular command and then pressing RETURN to take you back onto the editing area of the screen. The cursor cannot be moved into the stored command margin using the cursor keys, though: instead you use function key SHIFT-18, EDIT COMMAND. Do this and the cursor will move into the margin. Next type:

CE

and press RETURN. You have now entered the stored command to produce centred text, which we shall examine in a moment. Occasionally you will want to delete a stored command. This is done by moving the cursor to the line containing the stored command and then pressing function key SHIFT-19, DELETE COMMAND.

Some of the straightforward stored commands are discussed below, and we will describe others during the course of the book. Essentially, there are two types of stored commands. Some, such as CE, expect text to be entered on the same line, so that the command can then act upon it. Others require no text and affect the presentation of the text generally. With this second type, the rest of the line should be left blank as any text on it will be ignored.

CE - Centre Text

This stored command will centre the text following the command. It will be centred about the middle of the current ruler. To see how CE works, reload the LUNAR text saved earlier. Move to the line containing the heading for the piece, 'Tranquility Base', and enter the stored command CE as described earlier. The result should look something like figure 8.1.

The Times Sunday July 21 1969

CE Tranquillity Base

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Neil Armstrong became the first man to take a walk on the moon's surface today. The spectacular moment came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

Figure 8.1. The CE stored command in use.

The text is not centred in Edit mode, but will be when you preview or print it out. The line in question will be centred as shown in figure 8.2. Note that CE will only work on text on that particular line - if you require several lines of text centred then the CE stored command must be entered on each line.

The Times Sunday July 21 1960

Transportation Base

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Nell Armstrong became the first man to take a walk on the moon's surface today. The spectacular moment came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

Figure 8.2. The centred text.

LS - Line Spacing

Letters and small documents are normally printed out line by line. However, when producing copy for editing, such as drafts of documents, reports and the text for this book, it is more convenient to leave extra spaces between each line, to give room for editorial changes and comments to be made. In such cases the line spacing needs to be altered. By default, VIEW assumes single line spacing, so - although no stored command is needed to start initially - it would be obtained with:

150

LS is entered into the stored command margin in the normal manner - RETURN is pressed and then the 0 is placed in the edit area. The command LS 0 may seem to be wrong to obtain single line spacing, but if we think of the command LS 0 as being read as 'no line spaces' or 'no extra lines' it takes on more meaning. Professional copy for editing is normally supplied using double line spacing, ie, inserting a single blank line between each line of printed text. The stored command for this would be:

151

Figure 8.5 shows how the Edit mode screen looks using this command on the LUNAR text, and its effect when previewed using the SCREEN command.

P 1 = "The Times Sunday July 21 1969
LS 1
CE Tranquility Base
The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The final word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".
Neil Armstrong became the first man to take a walk on the moon's surface today. The spectacular moment came after he had finished his say down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

Figure 8.5a. The LS1 stored command.

The Times Sunday July 21 1969

Tranquility Base

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base, The Eagle has landed".

Neil Armstrong became the first man to take a walk on the moon's surface today. The spectacular ascent came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

Figure 8.5b. The LS 1 stored command's effect.

The value of LS can be anything up to 255, though this would not be of much practical use. To revert to normal single line spacing at any point in the text you simply use the LS 0 stored command as described. You can have as many LS commands in a document as you wish. The command will produce the desired effect up until the next LS command or when the end of the document is reached, so it is possible to have a variety of spacings in a single document.

Margins

The ruler regulates the current width of the text. At the far right-hand side of the ruler is the right-hand margin stop indicated by the < symbol. With formatting enabled, when the current word passes the position of this character the word is moved down onto the next line. There is no such left-hand character margin stop on the default ruler, because the left-hand side of the ruler itself will normally provide the margin. However, a left-hand margin stop can be used by editing a default ruler to include the > character. To see how it will affect text, clear any text from VIEW and then edit a default ruler as illustrated in figure 8.6.

.....>..........*.....*.....*.....*

Figure 8.6. A default ruler with left and right margin stops.

Now notice where the cursor sits, directly to the right of the left-hand margin stop. Any text you type from now on will be formatted between the two stops. Figure 8.7 shows the effect on the LUNAR text.

[illegible]

The Times Sunday July 21 1969

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Nell Armstrong became the first woman to take a walk on the moon's surface today. The spectacular event came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

.....

Figure 8.7. Margins allow narrow columns of text.

Move the cursor to the last line of the text and press RETURN. Now move the cursor to the left with the left arrow cursor key. The cursor will move until it comes to the stored command margin. Now enter the following text:

Meanwhile Michael Collins orbited the moon in the Command Module.

The text flows directly across the screen and ignores the > stop character - until that is the text is formatted using function key f0, at which moment the text will be rearranged within the bounds of the > < characters.

One of the reasons for wishing to arrange a left margin as shown is to enter text with side headings as in figure 8.8.

[illegible]

Figure 8.8. Using margins for side headings.

In this instance however, formatting the text will not alter the layout. This is because the side headings have been placed in the left-hand margin by first releasing the margin setting using function key SHIFT-12, MARGINS. The technique is as follows:

- * Press SHIFT-I2 to move the cursor into the margin.
- * Type the text (ie, CE) and press the right arrow key to move the cursor back to the > character.
- * Move the cursor up onto the line and enter text.

As with rulers, in general margin settings can be arranged and rearranged within a document as many times as you please and each one will only affect the text below it, up to the end of the document or the next ruler. Figure 8.9 overleaf shows an example in Edit and then Preview modes of a short section of text using several margin settings.


```

P.....
This is some sample text to illustrate the use of
margins within VIEW.
.....
This is some sample text to illustrate the use of
margins within VIEW.
.....
Text can be shifted across to the right
hand side of the screen with little
trouble.
.....
If you prefer text hand be limited
to the left side of the screen.
.....
Note 1:      Or the centre of the screen using the
              left margin for headings just in the way
              shown here!
.....
Pressing TAB moves
              the cursor to the first
              TAB stop in the main edit
              area.
Pressing SHIFT-F2 and
then TAB will move to TABs in the margin.

```

Figure 8.9. Default rulers with margin settings.

Beyond the Right Margin

Text may be placed beyond the right hand-margin. This is useful when making notes on text, for example. To allow this to take place, formatting must be turned off. Unlike the left, the right margin has no protect mechanism, so any subsequent formatting of this paragraph will ruin the layout as the text beyond the right margin will be brought back within the bounds of the main ruler and arranged accordingly.

Margin Stored Command

There is a single stored command that enables a margin to be set at the time of preview and printing - the command is LM, Left Margin. It can be thought of as a way of setting the left-hand margin stop. Consider the stored command:

LM 5

When the text is printed or previewed, the left-hand margin will begin five character spaces to the right. Note that the line of text does not become five characters short - it is all merely shifted

rightwards by five characters. The LM command will affect the text up until the next LM command or the end of the text. Figure 8.10 shows the LUNAR text once again with two LM commands in use. The first comes immediately before the first paragraph and sets LM 5, while the second paragraph is set to LM 10.

The Times Sunday July 21 1969

CE Technology Case

105

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

10 10

Nell Armstrong became the first man to take a walk on the moon's surface today. The spectacular ascent came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

The Times Sunday July 21 1969

Trough/114 Base

The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility base. The Eagle has landed".

Nell Armstrong became the first woman to take a walk on the moon's surface today. The spectacular ascent came after she had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched his movements from inside the craft.

Figure 8.10. LM stored commands in the LUNAR text (top) and the resultant output when previewed (bottom).

9 : Page Layout

In almost all wordprocessing you will be aiming to transfer your final draft document onto a sheet of paper - the hard copy. Being able to fit text within the dimensions of the paper and arrange it neatly is important. VIEW has a variety of stored commands which allow you to adapt and alter the way in which text is printed on the page - the page layout. To use VIEW to obtain a professional result relies not only on your literary skills but also on your ability to use these stored commands effectively.

The Default Page

By now you will almost certainly have attempted to print something prepared with VIEW. So far we have not discussed presentation of text: VIEW has always arranged it for us in a particular way. VIEW has a built-in set of default page layout settings - a standard page if you like - but they can easily be changed. The standard page is arranged as follows, working from the top down:

- * 4 blank lines - this is the Top Margin
- * 1 line - the Header line into which a heading can be placed
- * 4 blank lines - this is the Header Margin
- * 48 text lines - this is where your text is printed
- * 4 blank lines - this is the Footer Margin
- * 1 line - the Footer line into which page numbers and footing text can be placed
- * 4 blank lines - this is the Bottom Margin

By totalling these up we see that each page actually consists of 66 lines, 16 of which are blank, 2 of which can contain heading and footing text and 48 of which will carry the document. We can alter any one of these as we wish by using a stored command. The relevant commands are as follows:

PL - Page Length
TM - Top Margin
HM - Header Margin
FM - Footer Margin
BM - Bottom Margin

This layout and the stored commands which will alter each setting are shown in figure 9.1, and are discussed more fully below, once we have looked at headers and footers.

	TM 4 blank lines
VIEW: A Dabhand Guide by Bruce Smith	Header 1 line
	HM 4 blank lines
<p>The regular text of the page starts here and goes on to the bottom of the 48 line text area...</p> <p>... in other words up to here.</p>	Text area 48 lines
	FM 4 blank lines
Chapter 9 Page 107	Footer 1 line
	BM 4 blank lines

Figure 9.1. The default page layout.

Headers and Footers

Look at the top and bottom of this page, and indeed most other pages in the book. At the very bottom there is the page number and at the top the book's title. Most other books and even magazines and reports also have this. The line at the top of the page is called the header and that at the bottom is the footer.

Footers

Figure 9.2 shows a the bottom of a typical printed page. The numbers on the right refer to line numbers. The footer is printed in the footer line as described in the last section, and in a default page layout this is the fifth line from the bottom of the paper or the 62nd line on the page. The footer in this instance relays the word Page and the current page number. Line 57 contains the last line of text and the page reaches its end at line 66.

```

56:      This is one from the last line on a default page
57:      This is the last line of text on a default page
58:
59:
60:
61:
62:                      Page 23
63:
64:
65:
66:      Page ends here

```

Figure 9.2 The bottom of a default page.

VIEW will not print a footer automatically: unless you tell it to do so it will print a blank line. To define a footer we use the stored command DF. As an example, load the LUNAR text and insert a new line at the top. Enter the stored command DF and press RETURN. The footer information must be entered onto the line containing the DF command - enter the following:

```
DF      Page
```

ESCAPE into Command mode and preview the text with the SCREEN command. As the page finishes the word Page will be printed in the footer line on the right of the screen. In fact this would be printed on every single page of your document should it be extended beyond a page. To enable the printing of page numbers we must edit the DF line as follows:

```
DF      Page |P
```

The 'word' |P is a stored command (even though its not in the stored command margin) which tells VIEW to print the current page number, of which VIEW keeps a 'mental' note. Now preview the text. The page number is printed, but in the middle of the page with the word Page to the extreme left. The DF stored command allows us to range text into three possible positions. The syntax of the DF stored command is:

```
DF /left/centre/right/
```

The slashes are important, as by counting these VIEW knows in what part of the footer line to place the text, ie, to the left, to the right or in the centre. To print 'Page' and the page number in the centre of the page the DF line should be edited to read:

```
DF //Page |P//
```

Similar to justify the page details to the right or left we would use the DF command respectively as:

```
DF /Page |P///
```

```
DF ///Page |P/
```

We are not limited to entering just a single item of information - we can place text into any of the footer compartments as follows:

```
DF /View: A Dabhand Guide//Page |P/
```

Here the centre of the footer will be blank, whereas the footer defined as:

```
DF /View: A Dabhand Guide/Bruce Smith/Page |P/
```

will have details printed right across the bottom of the screen.

Headers

Headers are defined to the same principles as footers, but using the stored command DH. The header will be printed in the Header line which in a default page setting is the fifth line as shown in figure 9.3.


```
Page starts
01:
02:
03:
04:
05: Header details
06:
07:
08:
09:
10: Text starts on the tenth line in a default page layout
```

Figure 9.3. Position of a header.

To print a book title plus the author's name on the top of each page, the following header could be used:

```
DH /View: A Dabhand Guide//Bruce Smith/
```

Placing this header definition at the start of a document means that it will also be printed on the very first page of the document. In many cases this is not desirable as the document's title will be probably be printed at the start anyway. To avoid this, the header should be defined after the first line of printed text. It will then miss the first header line but is in position to be included on the second header line.

Headers and footers can be redefined at any time within your text. Once a header or footer has been re-defined it will be printed when the next header or footer line is reached. For example, when writing magazine copy it is useful to define a footer with the word 'MORE....' on the bottom of each page. On the last page this footer would be incorrect so I place a footer containing the word 'ENDS' at the bottom of the text.

On and Off

Although we may wish to define headers and footers we may not wish to have them printed on every single page so they can be turned on and off using an appropriate stored command as follows:

```
HE on/off    - turn header on or off
FO on/off    - turn footer on or off
```

If either is turned off then VIEW will simply print a blank line in the relevant position until the facility is turned on again. The

```
HE off
```

command could be used to ensure that the header is not printed on the first page of a document rather than using the method described

earlier - of course it will need to be turned on before the next page of the document is printed.

Positioning the Footer

Unless specified otherwise, the footer will be printed on the 62nd line of a page with margins of 4 blank lines above and below. By assigning new values with the footer margin and bottom margin stored commands it is possible to set the position of the footer in any of the bottom 9 lines of the page. If we wished, the footer to be placed on the very last line of the page, while keeping the same number of lines of printed text. We must use:

```
BM 0
FM 8
```

The last line of text will still be printed on the 57th line but the footer will now be printed on the 66th and last line. Eight blank lines is rather wasteful so we may wish to print text so that the footer remains on the 66th line but the text ends on the 64th line. This can be done by setting the footer margin to 1 and the bottom margin to 0:

```
BM 0
FM 1
```

Positioning the Header

The position of the header can also be altered in a similar fashion by use of the stored commands:

```
TM      Top Margin
HM      Header Margin
```

By adjusting these settings along with those of the footer, it is possible greatly to lengthen the number of printed lines per page. To print a header on the top line (first line) of a page, leave a blank line and then begin printing text on the third line we would use:

```
TM 0
HM 1
```

Setting BM 0 and FM 1, we would end up with headers and footers plus 62 lines of printed text per page.

Page Length

All of the above adjustments will result in more printed text per page, but of course by increasing the values of TM, BM, FM and HM it

is possible to reduce the number of text lines per printed page. However, none actually alter the length of the page itself - 66 lines by default. This is just right for continuous printer stationery, otherwise known as listing paper.

But the above settings are certainly not correct for standard headed paper, which is A4 size. In such cases you will need to adjust the overall printed page length using the PL command. Experiment with the PL setting to suit your own paper length. A4 size paper will normally require a PL setting of 70 as follows:

PL 70

Sizes of headers, footers and their associated margins can then be adjusted until you get the formula that is correct for you. Calculating the number of printed lines per page is quite easy - just add up the size of all the margins and subtract the value from the page length. For example,

TM	=	0
HM	=	1
FM	=	1
BM	=	0
Header line	=	1
Footer Line	=	1
TOTAL	=	4
Page Length	=	70

The total lines printed per page would be calculated as

Page length - (margins totals + header and footer)

With the above settings this would work out as:

$$70 - 4 = 66$$

To give 66 lines of document text per page.

When you have found your standard page it is well worth saving a copy of this to disc under a suitable file name such as STANDARD, which can then be loaded in as you prepare each document. Figure 9.4 shows a suitable set up dummy document, the contents of which could be loaded automatically by your !Boot file by adding a LOAD command to the end of it as follows:

```
*WORD
MODE 3
SET FJI
LOAD STANDARD
```


[illegible]

Figure 9.4. A standard page.

Page Ejecting

In the normal course of events VIEW will automatically eject a page when it comes to its end. This page ejection takes the form of the footer and bottom margin being printed on the current page and the top margin and header on the next page. On occasions however, it is useful to be able to 'force' a page to be ejected even though that page is not full of text. For example, Section 2 of a document might begin half-way down a page, but it would be more presentable if it were to begin on a new page. By inserting the stored command PE at the end of Section 1 (immediately before the start of Section 2) the page will be ejected and Section 2 will begin on a new page.

The page eject command may also be a conditional one. Suppose we have a table that is 10 lines in length including title. If the table was towards the end of some text, it is quite possible that a page break would occur some way through the table, splitting it across two pages. This may not be desirable. In such cases, specifying a number after the PE stored command will force a page eject should there be fewer lines remaining on the page than the number specified, thereby ensuring that the text or figure after the command is placed on a new page. In the case of our 10 line table we would use the stored command:

PE 10

so that if there are fewer than 10 lines of text space left on the page, it will be ejected. The PE comand does not affect the printing of the footer - this will be printed as normal unless FO OFF is used.

Normally, VIEW will eject to the start of the next page. It is possible, though, to force VIEW to eject to the next odd or even numbered page using the stored commands:

- OP - eject to next odd numbered page (ie, 3,5,7 etc)
- EP - eject to next even numbered page (ie, 2,4,6,7 etc)

Uses for both these stored commands may not be immediately obvious so here is an application for each. If you have a two page spread of tables then it would be more effective to ensure that the two pages form a spread or 'landscape'. To allow this to happen the first page of the tables must be printed on an even number paged, using EP before the first table will ensure this. A use for OP would be to ensure that each new chapter or section head appears on an odd (right hand) page by using it before the page is printed.

Paging and Page Numbers

VIEW splits a document into pages, inserting page breaks, headers and footers at the appropriate points. However, we can disable the paged facility so that the document is printed as a continuous sheet without any breaks, footers and headers using the page break stored command PB as follows:

- PB OFF - page breaks OFF
- PB ON - page breaks ON

With page breaks ON the current page number can be printed out anywhere within a document using the special stored command IP. It is special in that it does not go into the stored command margin but instead works in conjunction with DH, DF, RJ, LJ and CE.

As your computer's memory is limited, long documents will often have to be split into a number of smaller sub-documents. In such instances page numbers will be affected - by default each sub-document will start with page number 1, and in a 50 page document this can be confusing!

To overcome this difficulty, VIEW has a stored command which allows us to set the value at which IP begins: it is SR. Suppose we have finished part 1 of the document - we preview this with SCREEN and note that the last page number is 12. After clearing the existing text, we start editing part 2 of the document and so would wish page numbers to start at 13 - using the SR stored command at the very start of the text we would enter:

SR P 13

and now the first page printed will be numbered 13, the second 14 and so on. Note that only SR is entered into the stored command margin - the P and 13 are within the edit screen but on the same line as the SR stored command. If we need to use a third or fourth part then the procedure is exactly the same: note the last page number and use SR to set it to one greater in the next part.

Look at the bottom of any two pages in this book. The page numbers are always on the edge of the page. This does not happen by accident – they are placed there for ease of reference. If the page numbers were always printed in the same position then half of them would be lost in the book spine. The numbers could be placed in the centre of the footer, but this makes it slower to flick through to locate a particular page. This is not only important with footers; with headers, we might want the document name printed on the outer edge of the page as well. Again, VIEW can handle this automatically. The stored command is TS (two sided):

TS	ON
TS	OFF

With TS ON, VIEW will alternate the footer and header details for you. TS OFF turns the facility off, and is the default state. Try experimenting with some text to see exactly what happens - figure 9.5 shows a VIEW screen making use of the TS stored command.

```

01 *****
LN 5
OH /Lunar Text//Ban on the Moon/
OF ///Page 1P/
TN 1
OM 1
HM 1
FM 1
PL 16
TS OH
The Times Sunday July 21 1969

```

CE Tranquility Base
The landing in the Sea of Tranquility, was near perfect and the two astronauts on board Eagle reported that it had not tilted too far to prevent a take-off. The first word from man on the moon came from Aldrin: "Tranquility Base. The Eagle has landed".

FE
Neil Armstrong became the first man to take a walk on the moon's surface today. The spectacular moment came after he had inched his way down the ladder of the fragile lunar bug Eagle while colleague Edwin Aldrin watched him

Figure 9.5. The TS stored command.

Note the positions of headers and footers. In this instance the header and footer are defined as follows:

```
DH /Lunar Text//Man on the Moon/  
DF ///Page 1P/
```

When this page is printed the headers and footers on page 1 and 2 will be printed in position as follows:

Page 1:

```
Lunar Text      Man on the Moon  
                  Page 1
```

Page 2:

```
Man on the Moon      Lunar Text  
Page 2
```

We have already encountered the left margin stored command, LM. The TS ON command also allows us to follow it with a margin indent value to take into account the 'gutter' which is found in all books. The gutter is the margin area which goes into the binding of the book - it effectively spreads across the 2 pages which face each other. The stored command:

```
TS ON 10
```

would provide a gutter margin of 10 characters on each page - the margin would be switched to the other side of the page when the next page is printed and back once again on the third page and so on, ensuring that the margin is always facing to the inside of the text.

10 : Macros

At the very start of this book we saw how a wordprocessed letter can be used to mail large numbers of people about a particular item or offer. A couple of chapters back we also saw how the CHANGE command could be used to adapt the letter, changing names, addresses and so forth. However, producing perhaps several hundred letters in this way would be time consuming. Instead it is almost certain that a particular form of letter mailing was used, handled by a macro. The word macro in wordprocessing simply means a way of repeating a particular task quickly and efficiently.

The VIEW macro can be thought of as a large stored command and the letter to be printed as a template in which 'holes' are left. Into these holes are dropped the name and address of the person concerned, and any other information we may wish to include. So that VIEW knows where to insert the information, a special character is used - this is the @ character. The @ is followed by a number, the first of which is always 0, so the first three macro symbols would be @0, @1 and @2. Now we need to assign to each of these macro symbols some information, so we could say in our initial example that:

@0 = Mr B Keeper
@1 = 103 Acacia Ave
@2 = Arkley

Now as VIEW prints the letter template, whenever it comes across @0 it will print 'Mr B Keeper'. Similarly when it encounters @1 it will print '103 Acacia Ave' and 'Arkley' when it gets to a @2. By redefining the information assigned to @0, @1 and @2 another letter can be printed and the new information 'dropped' into the template as it is printed, and of course this can go on for as long as we wish, or rather for as long as the information is supplied.

That's the theory, so let's try a simple macro exercise for ourselves - and I do suggest that you work through this one!

A macro is defined using the stored commands DM and EM. DM is define macro and so comes at the start, whilst the macro is ended by EM. Macros are called by name - the name however is limited to two letters which may be any combination, such as AA, BB, AB, ZZ, SD. It

is important to remember that character case is significant so that a macro called aa is different to one called AA. The macro name must follow the DM command, ie:

07 99

Figure 10.1 shows how your macro should be entered into VIEW.

[illegible]

Figure 10.1. A simple macro setup.

We have included the PE (page eject) stored command within the macro - in fact we can include any of the stored commands that are suitable. The macro is thus defined, and now we must provide the information for it to work on. Press function key SHIFT-F7 EDIT COMMAND to move the cursor into the stored command margin. Now you enter the two letters that make up the macro's name, AA in this case, and press RETURN. On the same line the items we wish to be assigned to each character are entered, each being separated by a comma, thus:

AA Mr B Keeper, 103 Acacia Ave, ARKLEY, Hertfordshire

Spaces within the parameter line, as it is known, are significant, so placing a space after a comma within the line will result in a space being printed. To see how the macro turns out we can either print it or preview it with the `SCREEN` command, and the effect is shown in figure 10.2. Each @ character within the template has been substituted by its assigned value from the parameter line.

Mr B. Keeper
103 Acacia Ave
BAXLEY
Hartfordshire

Figure 10.2. The macro in action.

We can now add another parameter line if we wish, in fact as many as we want, and they all follow the above format. After the last parameter line add another one as shown below, remember that the AA should sit within the stored command margin:

AA Mr D Abs, 76 Gardner Road, Prestwich

Now when you print or preview the macro two pages will be displayed/printed, the second taking and showing the details of Mr D Abs.

Figure 10.3 shows part of the layout of the original letter in Chapter 1 and how it might look in macro form. The format of the letter looks untidy, but it will be squared up when the template details are added. Presentation is worth bearing in mind when designing a general purpose macro. Previewing the layout and use of the function keys to split and join lines will get the final layout to an acceptable form.

[illegible]

Dear Sir,
Because of the expansion of our company, we are considering opening a showroom in the W2 area. To help promote our company in this area, we are looking for a select number of homes into which we can install our full and comprehensive range of windows, doors and patio doors.

Figure 10.3. Macros in the original letter.

The length of total parameters within a macro is limited only by the number of characters you can type onto a line - in VIEW this is 132 characters, and as with all stored commands they are not kept to the limit specified by the current ruler, ie they may go beyond it.

The parameters within the parameter line are separated from one another by use of a comma. So what if we wish to include a comma in

the parameter itself? For example, when sending a letter to the firm of:

Fox, Fox and Fox

The comma after the first Dickens would be the end of the first parameter. The solution: if you wish to include a comma within a parameter you include the text around it inside angled brackets thus:

<Fox, Fox and Fox>

The parameter line might then be:

AA <Fox, Fox and Fox>,33 The Larch,Stafford

Number Registers

In the last chapter we saw that VIEW keeps track of the current page, which can be printed within a stored command by use of the command IP. P is a special location within VIEW known as a register. There are 26 registers in all, labelled from A to Z. Four registers other than P are used by VIEW, though two of these are limited to the Master 128 and the Master Compact. The five registers used are:

P	Page number
C	Chapter number
L	Line count
D	Date (Master 128 & Compact only)
T	Time (Master 128 & Compact only)

The C register is used to hold the current chapter number. Unlike P, which keeps track of the current page number for itself, C must be set manually each time a new chapter is started. All registers are set with the SR command in the same manner. To set Chapter 1 at the start of a document use:

SR C 1

To print 'Chapter 1' on the first page a suitable command would be:

CE Chapter IC

and of course it could be included in headers and/or footers:

DF /View: A Dabhand Guide/Chapter IC/Page IP/

To increment the C register by 1 each time a new chapter starts, use the stored command:

SR C IC+1

The current value of C will then have one added to it. The format of this command is the same to increment any register.

A count of lines per page is kept in the L register by VIEW - VIEW itself automatically increments this register each time it 'prints' a line. It is set to 0 when a page break occurs. Like all other registers it can be used in a stored command:

```
CE Current Line count is: |L
```

The T and D registers can only be used on the Master 128. They are ideal for time and date stamping a letter. Letters are always dated so on a Master 128 you could set up a letter heading like this:

```
Dabs Press
76 Gardner Road
Prestwich
Manchester
RJ 10
```

The D and T commands can be used with the VIEW supplied with the Master Compact, but the clock on this is static (unless you have Econet) and will only ever print the date and time fixed at:

```
31 Dec 1999, 23:59:59
```

So I suppose you could use the facility once!

Chapter Sub-headings

Many documents and books use a section numbering system such as 2.3.1 which would be chapter 2, section 3, subsection 1. For example:

```
10.1 Introduction
  10.1.1 Scope
  10.1.2 Aims
  10.1.3 Notes
10.2 How to Write
  10.2.1 Style
  10.2.2 Layout
```

We can quite happily print out the chapter number at the start of each section using the C register. To obtain the section number and subsection numbers, we set up two unused VIEW registers to keep a count of them. As an example we decide to use registers A and B as follows:

```
A = Section numbers
B = Subsection numbers
```


To print the first heading ('10.1 Introduction') we must set register C to 10 and register A to 1 as follows:

```
SR C 10
SR A 1
```

A separate stored command must be used for each and these might be printed on the left of the page with the LJ stored command:

```
LJ |C.|A Introduction
```

Subsections 10.1.1 through to 10.1.3 require no alteration to be made to registers C or A but register B must be first set and then incremented from 1 through to 3 as follows (the comments on the left in brackets are for your information only and should not be entered):

```
SR B 1 (Set register B=1)
LJ |C.|A.|B Scope (Print chapter, section and subsection)
SR B |B+1 (Increment register B by 1)
LJ |C.|A.|B Aims (Print chapter, section and subsection)
SR B |B+1 (Increment register B by 1 to 3)
LJ |C.|A.|B Notes (Print chapter, section and subsection)
```

When this is printed or displayed with SCREEN the registers' values will be printed to give:

```
10.1 Introduction
  10.1.1 Scope
  10.1.2 Aims
  10.1.3 Notes
```

To print the items in section 2 of chapter 10 the A register needs to be incremented to 2 and then the subsection register B set to 1 and then 2. this would be done as follows:

```
SR A 2 (Set register A=2)
LJ |C.|A How to Write (Print chapter and section)
SR B 1 (Set register B=1)
LJ |C.|A.|B Style (Print chapter, section and subsection)
SR B |B+1 (Increment register B by 1 to 2)
LJ |C.|A.|B Layout (Print chapter, section and subsection)
```

Printing this will display:

```
10.2 How to Write
  10.2.1 Style
  10.2.2 Layout
```

Of course, all the above might at first sight seem a convoluted way to number sections and subheadings, and of course you are right. However it has two advantages. The first and most obvious is that should you decide to rearrange your document so that chapter 10 moved to become chapter 12 and you also decided to insert an extra subsection here and there, altering the numbering would only require a few stored command settings to be altered rather than having to go through the entire document and alter every single occurrence. The second big advantage comes when using registers with macros, and we will examine that in the chapter after next, after we have examined the somewhat thorny task of printers and printing text.

11 : Printers & Drivers

Text in VIEW can be sent to a printer simply by entering Command mode and typing:

SHEETS

This was discussed briefly in the last section of Chapter 3.

The document in memory will be printed in the standard default character set of the printer. However, most printers these days support extra printing facilities. Virtually all machines will allow you to underline text, and to embolden it by printing each character twice to provide a darker print. These two effects are ideal for document headings. Other effects are also possible depending on the type of printer you have: italics, double height, double width and so on. Your printer manual contains details.

The effects can be reached from VIEW. However, to be able to use them a special form of program called the printer driver must be loaded into VIEW. By inserting special codes (called highlight codes) into the text of your document it is possible to produce these extra effects.

Printer Types

Essentially there are two kinds of printer. The most popular type is the dot matrix machine. Epson, Star, Citizen and Taxan all manufacture good examples. The printer 'constructs' each character as it needs to be printed by pushing out a grid of small pins formed into the character shape. The pins strike the paper through an ink ribbon, producing the character. The second type is the daisywheel printer. This is rather like an electronic typewriter in that it contains a wheel of characters which is rotated to the desired character and then printed.

For producing high quality text to a professional standard, the daisywheel printer is best. But it is often slow, and can only produce a very small range of printing effects, bold and underline being the most common. Different styles of text can be produced by changing the

daisywheel itself. Without a doubt the dot matrix printer is more versatile - and indeed all the latest of these enable you to select a near letter quality (NLQ) effect, which is acceptable for letters and documents. Some dot matrix printers even support letter quality (LQ) print, which can be almost indistinguishable from daisywheel output.

If you are going to buy a dot matrix printer then there is one golden rule: it must be Epson-compatible. The Epson printer, and in particular the Epson FX80 printer, has been adopted as a sort of unofficial world standard and the codes that control the special effects have been carried across to most but not all dot matrix printers. An Epson-compatible printer then is one that supports and will recognise all of the Epson effect codes (referred to as control codes). This does not mean that you should rush out and buy an Epson printer; on the contrary, you should buy one that has any extra facilities that you want, but it should support the Epson standard set and at least have an NLQ capability.

The Printer Driver

To create the special program that drives the printer to obtain special effects, you will need to use a program called a printer driver generator (PDG). The official Acornsoft version can be obtained from any Acorn Dealer and includes an easy to follow instruction manual. The program disc which accompanies this book (see Appendix M) also contains a specially written PDG program which supports extra facilities.

Master Compact owners will find that their version of VIEW contains an Epson FX80 printer driver already installed so that if you are using this type of printer or a compatible model you can go straight ahead and do so, with no need for a PDG.

Free Bees

If you already have a printer driver then you can skip the next few paragraphs if you so wish and move onto the section marked 'Loading the Driver'. Having said that this new page or two does explain how printer control codes work and so you may find it worth reading if you are unsure to there operation.

Listing 11.1 provides the simple printer driver which will allow you to select two effects from View - the choice is yours but the most

useful are bold or underlined text. To make use of this program and the more sophisticated Printer Driver Generator program on the Programs Disc it is important that you understand just how to control your printer. This can be an off putting aspect of wordprocessing but I assure you that there is no particular magic involved and if you read what follows carefully you will master printer control.

Control Codes

The printer has what can be called its own language. This language is in the form of numbers. When a printer receives a certain sequence of numbers it will act upon them. The telephone exchange recognises numbers you dial on the telephone and uses this to select one particular phone in the whole world. Use a different sequence of numbers and a different phone is called. Send a different sequence of numbers to the printer and a different effect is selected or perhaps a selected effect is cancelled. The numbers that are sent to the printer are called **control codes** or **Escape codes**.

To select the effect that you want from your printer in means looking at the printer manual to extract the necessary codes - not as painful as it seems. To help a bit you'll find that Appendix J contains a list of more important printer control codes - you can extract the codes you want from here or look through your printer manual.

Printer manuals will normally provide control code sequences in one of two ways, ie ESC or CHR\$ codes - most printer manuals have both, so concentrate on the row of CHR\$ codes. For example a typical Epson printer manual the table to select emphasised (bold) text might look like this:

CODE : ESC E

PURPOSE : Select emphasised print

FORMAT : CHR\$(27) CHR\$(69)

REMARKS : This command causes the printer to print in emphasised (bold) mode

All the information needed to produce emphasised text is here! The line we are really interested in is the line marked FORMAT,

FORMAT : CHR\$(27) CHR\$(69)

indeed all we have to do is to take the two numbers given in that line,

27 and 69

and place a 1 in front of each, so that the control code sequence to be sent to the printer becomes:

1,27,1,69

Sending this will turn bold print on an Epson compatible printer. You can try this from BASIC first with the following 5 line program:

```
10 VDU 2 : REM turn printer on
20 PRINT "This is normal text"
30 VDU 1,27,1,69
40 PRINT "This is bold text"
50 VDU 3 : REM turn printer off
```

If you find that your printer manual does not contain these CHR\$ values then it will certainly have the ESC character sequences, that you can quite easily convert into CHR\$ values. In the above example the italic font was also specified as

CODE : ESC E

ESC means ESCAPE. The ASCII code for ESCAPE is 27, similarly the ASCII code for 'E' is, yes you guessed it 69. So ESC E is 27,69.

Up to a point

Once a printer effect has been selected with an output control sequence the printer will continue to act on it until it is switched off or it is cancelled by another command. The manual entry to turn emphasised (bold) print off might look like this:

CODE : ESC 5

PURPOSE : Turn emphasised print mode off

FORMAT : CHR\$(27) CHR\$(70)

REMARKS : This command causes the printer to cancel the emphasised print mode

To cancel emphasised print then the control code sequence would be: 1,27,1,70

We could add 2 more lines to the BASIC program above as follows:

```
42 VDU 1,27,1,70
43 PRINT "Normal text once more"
```

Underlined Text

Text underlining is a useful capability and for this reason it is generally used as the 'other' effect on printer drivers. Not only is it useful for emphasis (as is bold) but it is also ideal for underlining section headings etc.

From a programming point of view, selecting underlined text provides a new challenge:

CODE : ESC - 1

PURPOSE : Turn underlined print mode on

FORMAT : CHR\$(27) CHR\$(45) 1

REMARKS : This command causes the printer to underline text

Unlike emphasised print, underlined print is one of several printer effects that require 3 control codes to be sent. The procedure is just the same and looking at the above table to get underlined print we must send each of the above codes (27,45 and 1) to the printer, remembering to precede each with a 1 thus:

1,27,1,45,1,1

The sequence to turn off underlined text is almost identical except that the final 1 becomes a 0 thus:

1,27,1,45,1,0

HomePDG

The HomePDG as I have called it uses the emphasised and underlined codes to produce these effects - you can see that these are placed in DATA statements towards the end of the listing itself, the only difference being that each control code sequence ends with a 255 byte to denote that fact. Obviously you can enter your own codes here if your printer is not an Epson compatible, or if you wish to use other effects. Remember though that you are limited to just two.

Once you have entered the program save it before running. The program does not include any self checking because it will vary from user to user, so do double check the listing. Once RUN the program

will automatically save the printer driver as "Edriver" and is ready to use.

A final note concerning the printing of pounds. Epson compatible printers have built into them several standard character sets, by default the American mode is selected which does not have a pound character. By setting dip switches within your printer (see your printer manual) then the English character set can be permanently enabled, and this of course contains a £ character. However the £ character shares an ASCII code with the hash, #, character so if you wish to print a pound then the # must be inserted into your text. Printer Driver Generators such as Acomsoft's and the one on the Programs Disc will allow you to configure your printer driver to print pounds regardless of dip switch settings.

Loading the Driver

Enter VIEW with the *WORD command, select Mode 3 and reset with NEW. Master Compact owners will have an Epson FX80 driver installed automatically, other users will need to load in the newly created driver. If you are using the DIY printer driver type the following command from Command mode:

```
PRINTER Edriver
```

The disc drive will come to life and the Edriver printer driver will be loaded. The PRINTER command is a special sort of LOAD command - it looks for a printer driver specified after the command PRINTER and loads it into the printer driver area. Once the driver is loaded it is ready to use.

If you already have another printer driver then simply load this in the same way using the driver's name, ie if the driver is called FLO16 then it is loaded with:

```
PRINTER FLO16
```

More Codes

Your printer will be able to produce many more special effects other than the two described above, especially if you have a dot matrix printer. You may wish to substitute another effect so here are some more that are possible on a standard Epson compatible printer.

Double sized or enlarged characters provide an excellent way to head the start of a document. A single printer code will select double sized text on Epson compatible, the code is 14 so the sequence is:

1,14

As you can see an ESC character is not needed. Turning double sized mode off is done with another single code, 20:

1,20

If you want to get more text on a line of printer paper then there are several ways to go about it, depending on how many characters per line you want. Pica-sized mode prints text at the rate of 10 characters per inch and is enabled with codes 27,70,1 so the complete sequence is:

1,27,1,70,1,1

Elite sized mode provides 12 characters per inch while condensed print squeezes in 17 characters per inch. The control codes for each are:

1,27,1,66,1,2

1,27,1,66,1,3

If you are processing documents that contain material of a mathematical nature then the ability to print superscript and subscript numbers will be most invaluable. Superscript mode is enabled with the output control sequence:

1,27,1,83,1,0

This can be used with most other effects printer with the exception of enlarged print. The code:

1,27,1,84

is used to cancel superscript and subscript printing. Of course it is possible to print entirely in superscript mode - suitable for producing the 'small' print that nobody ever reads!

Subscript mode works by printing text on the bottom of the line, as the following example clearly shows, the output code to enable subscript printing is:

1,27,1,83,1,1

Highlight Codes

To obtain a special effect we place a highlight code into the text - this is a special character which VIEW recognises as being a command to turn on a special effect until the highlight code is encountered again, at which time the special effect will be turned off. VIEW has two standard highlight codes, called highlight 1 and highlight 2 - these are obtained with function keys SHIFT-F4 HIGHLIGHT 1 and SHIFT-F5 HIGHLIGHT 2.

As only these highlight commands are available, we can only get at two of the range of effects directly. Selecting highlight 1 will make the printer produce underlined text while highlight 2 gives bold text. To try them out enter Edit mode and on the first line type:

This is normal text

On the second line first press SHIFT-F4 and you will see a white square with a small black dash in it (NB: in early versions of VIEW and when using Mode 7 you will see only a white dash). After this type:

This is underlined text

and then press SHIFT-F4. Again a white square with a black dash will appear and this will turn underlining off. If you fail to turn a highlight off then the effect will be continued throughout your text until the next highlight or end of text is reached. Chapter 18 contains a useful utility program which will check through a document to ensure that highlight codes 1 and 2 are balanced. On the third line do the same for highlight 2 enclosing them around the text:

This is bold text

highlight 2 appears as a white square containing a black asterisk (or just an asterisk in early versions of VIEW and when entering highlight 2 in a Mode 7 Edit screen).

Return to Command mode and type SCREEN. As the text appears you will see that both highlight codes are shown. Now with your printer attached and ready to go print the text by typing:

SHEETS

and pressing RETURN when the 'Page 1...' prompt appears. All being well the highlight codes should have the desired effect. Once the printing is completed return to Edit mode. It is also possible to use both highlight codes together - for example using highlight 1 and 2

at the start and end of some text will produce underlined bold text. Try it on the fourth line as follows, by pressing function keys SHIFT-14 or SHIFT-15:

Underlined and bold text

Note that an inverse minus sign means enter/exit highlight 1 and an inverse asterisk means enter/exit highlight 2.

Print this out to check that it works correctly.

The above examples all take effect on a single line of text - highlight codes can be used to embolden or underline anything from a single word through to a whole paragraph or document. The following example when printed would underline the words 'underlined' and print it and the rest of the line in bold.

Here is some **underlined** text while the rest of the non-**underlined** text is in bold only.

Extended Highlights

To gain access to those other special effects we need to employ an extended highlight, where we use a stored command to reset highlight 2 so that it sends a different code to the printer driver. When the printer driver sees this new code number it will allow the use of certain combinations of highlight 1 and highlight 2 to produce effects on the printer in addition to underlining and bold. The extra or extended effects are as shown in table 11.1:

Highlight	Effect
* - - -	Reset printer
-	Switch underlining on and off *
* * *	Switch bold on and off
* -	Begin subscripting *
* *	Begin superscripting *
* * -	Revert to normal print after super- or subscripting
* - -	Select alternative font (Pica) on or off



Switch italics on and off

Table 11.1. The extended highlight set.

The extended highlights marked with an asterisk are only effective for that line of text and will switch themselves off automatically should they not be cancelled on the same line, so if they are required on the next line then they must be re-selected. The reset printer highlight should always be used at the start of all documents to clear the printer of any effects that may have been left enabled from a previously printed document.

To use the extended highlight set it is necessary to reset highlight 2 using the stored command HT as follows:

```
HT 2 130
```

Once encountered the extended highlights can be used unless they are disabled by reselecting the normal highlight 2 code 129 with the stored command:

```
HT 2 129
```

Effects may be mixed so underlined text with bold italics is possible, as are other combinations.

Highlight codes are not limited to normal text. They may be included in stored commands including headers and footers if so desired.

SHEETS versus PRINT

The SHEETS command allows single pages of text to be printed. The page number is displayed before the page of text is sent to the printer. To print the sheet, any key is then pressed bar the M and Q keys. If M is pressed then that page will not be sent to the printer. Instead it will be shown on the screen and the next page in the document will then be prompted. In this way the SHEETS command can be used to print or reprint selected pages from a document. Pressing the Q key will quit printing.

VIEW also supports another printing command - aptly called PRINT. Entering this command will result in the whole of the document being sent to the printer without a pause between the pages, so is only suitable for use with continuous stationery. The PRINT command can

also be used to print VIEW files direct from disc without affecting the current contents of memory. For example, typing:

```
PRINT Chapter11
```

would look for a file called Chapter11 on the current filing system and print it - all highlight and stored commands are recognised and acted on. If several files are to be printed then a list of filenames can be included in the PRINT command, each separated by a space. The command sequence:

```
PRINT Chapter1 Chapter2 Chapter3 Chapter4
```

would send the four named files one after the other to the printer.

Microspacing

Justification has already been discussed. VIEW justifies text by inserting extra spaces between words to provide a straight right hand edge to the text of a document. While this leaves an even edge, the spacing of words within lines is often rather eccentric. Some printers provide a facility known as microspacing. Essentially, with microspacing enabled, VIEW divides all the extra spaces needed to justify a line of text (ie, those it needs to add itself) into divisional units of 1/120ths of an inch. Then it arranges these units evenly through the line by adding an equal number of them to each of the spaces within the line - thus ensuring that each space within the text is of an equal size.

If the MICROSPACE command is followed by a number then this determines the width of the characters, where n the number is measured in 1/120th of an inch - a default setting of 10 is taken if no number is specified.

If you are using the Acornsoft PDG it is possible to include this microspacing facility in the printer driver. If it is included it is enabled by typing MICROSPACE from command mode. The extended PDG on the Programs Disc (see Appendix M) does not support microspacing.

Text Formatting

The inclusion of highlight codes within text will give formatted text a ragged right edge on screen. However, text remains formatted as the highlight codes, although being sent to the printer via the printer driver, are not printed as characters.

Making a Hash of Pounds

It is a fact of life that virtually any printer you may care to buy will be set up ready to use for the American market - after all this is where more printers are sold than any where in the world. The relevance of this for us is that as it stands the printer will be unable to print the sterling pound sign, £. This is because the £ sign is not used very often and as such it does not appear in the printers American character set, in fact if you do try to print a £ you will get a ' instead. Virtually all printers have alternative character sets however and normally an English character set which will contain the £ sign of course. But the English character set does not normally contain an hash, #, sign. This can be annoying if you are using VIEW and listing assembly language programs as you will have to select the correct character set first, though this can be done by a !Boot file of course.

There are two ways in which a printers character set can be changed to the English font. By setting the printer dip switches so that the set is always selected or as mentioned via a BASIC command as part of your !Boot file. On Epson compatible printers the following sequence of VDU codes will select the English character set:

```
VDU2,1,27,1,82,1,3,3
```

So a very simple !Boot file to do this and then drop into VIEW with a printer driver installed could look like this:

```
*BASIC
VDU2,1,27,1,82,1,3,3
*WORD
MODE 3
NEW
SETUP FI
PRINTER Epson
LOAD letter
```

Note that BASIC is selected first before performing the VDU sequence and dropping us back into VIEW.

Having done this the £ sign will not normally be printed when you have a £ in your text. The £ sign has now taken the place of the # character in the character set. Thus placing a # in the text will cause a £ to be printed when it is encountered. This is unless you are using a full printer driver in which case when the Printer Driver Generator asks which character will produce a £ you should respond with 35

which is the ASCII code for a hash. Now you can use £'s in your text and when the driver encounters one it will send a # to the printer instead and print a £ sign. (You don't know the problems we had getting these last few paragraphs right coming from VIEW, via another computer, to typesetting!)

To reselect the American character set you can either turn your printer off and on or from BASIC type:

```
UDU 2,1,27,1,82,1,0,3
```

Listings

Listing 11.1.

```
10 REM HomePDG
20 REM (C) Bruce Smith 1987
30 REM VIEW: A Dabhand Guide
40 REM Highlights 1,2 and £'s
50 :
60 vector=&A8
70 oswrch=&FFEE
80 PROCassemble
90 *SAVE Edriver 5000+C0 400 400
100 END
110 :
120 DEF PROCassemble
130 FOR pass=4 TO 7 STEP 3
140 P=&400:O=&5000
150 [OPT pass
160 JMP print
170 JMP on
180 JMP off
190 JMP none
200 JMP none
210 :
220 .none
230 RTS
240 :
250 .on
260 LDA #2
270 JMP oswrch
280 :
290 .off
300 LDA #3
310 JMP oswrch
320 :
```



```
330 .print
340 STA acc
350 STX xsave
360 STY ysave
370 CMP #128
380 BEQ do1
390 CMP #129
400 BEQ do2
410 CMP #ASC"L"
420 BNE notpound
430 LDA #35
440 .notpound
450 JSR @FFE3
460 RTS
470 :
480 .do1
490 LDA boldflag
500 BMI turnboldoff
510 EOR #255
520 STA boldflag
530 LDA #boldon MOD 256
540 STA vector
550 LDA #boldon DIV 256
560 STA vector+1
570 BNE decodes
580 .turnboldoff
590 EOR #255
600 STA boldflag
610 LDA #boldoff MOD 256
620 STA vector
630 LDA #boldoff DIV 256
640 STA vector+1
650 .decodes
660 LDY #255
670 .codeout
680 INY
690 LDA (vector),Y
700 CMP #255
710 BEQ finished
720 JSR @FFE2
730 JMP codeout
740 :
750 .do2
760 LDA underflag
770 BMI turnunderoff
780 EOR #255
```

VIEW : A Dabhand Guide

```
790 STA underflag
800 LDA #underon MOD 256
810 STA vector
820 LDA #underon DIV 256
830 STA vector+1
840 BNE decodes
850 .turnunderoff
860 EOR #255
870 STA underflag
880 LDA #underoff MOD 256
890 STA vector
900 LDA #underoff DIV 256
910 STA vector+1
920 BNE decodes
930 RTS
940 :
950 .finished
960 LDA acc
970 LDX xsave
980 LDY ysave
990 RTS
1000 :
1010 .acc BRK
1020 .ysave BRK
1030 .xsave BRK
1040 :
1050 .boldflag BRK
1060 .underflag BRK
1070 :
1080 .boldon
1090 EQU 1
1100 EQU 27
1110 EQU 1
1120 EQU 69
1130 EQU 255
1140 :
1150 .boldoff
1160 EQU 1
1170 EQU 27
1180 EQU 1
1190 EQU 70
1200 EQU 255
1210 :
1220 .underon
1230 EQU 1
1240 EQU 27
```

```
1250 EQUB 1
1260 EQUB 45
1270 EQUB 1
1280 EQUB 1
1290 EQUB 255
1300 :
1310 .underoff
1320 EQUB 1
1330 EQUB 27
1340 EQUB 1
1350 EQUB 45
1360 EQUB 1
1370 EQUB 0
1380 EQUB 255
1390 ]
1400 NEXT
1410 ENDPROC
```

Listing 11.1. The HomePDG.

12 : Macros Revisited

The real versatility of macros comes by combining them with the PRINT command. For example, it is possible to write and have on disc a complete set of macros which can then be used to effect on a separate file of text by using a command such as:

```
PRINT Macro Text1 Text2
```

where Macro is the name of a file containing the macro definition(s) and Text1 and Text2 are the files to be printed.

A Simple Example

Let's work through a very simple example to see how a file macro can be used to influence a text file. The example uses a macro to increment a chapter number and read in a contents file to provide a chapter list of the first few chapters of this book in the form:

```
Chapter X - Chapter Title
```

Enter VIEW, type NEW and then enter the following stored commands which make up the macro called AA. Note that the bracketed comments at the end of each line should not be entered. These are for information only:

```
SR C 0           (Set chapter count to 0)
DM AA           (Start definition of macro AA)
SR C |C+1       (Increment chapter number)
LJ Chapter |C - @0 (Print 'Chapter' and number and text)
EM             (End macro definition)
```

This macro should now be saved to disc. Call it MacroAA and enter:

```
SAVE MacroAA
```

from Command mode.

Clear the macro from memory with NEW and then enter the following stored commands:

```
AA The Wordprocessor
AA Introducing Command and Edit Mode
```

```
AA The Ruler
AA Saving and Loading Text
```

and save this as as TextAA:

```
SAVE TextAA
```

To see the result we can print the two files. This can be done to a printer with the command:

```
PRINT MacroAA TextAA
```

or directly to the screen with:

```
SCREEN MacroAA TextAA
```

The following should be printed or displayed:

```
Chapter 1 - The Wordprocessor
Chapter 2 - Introducing Command and Edit Mode
Chapter 3 - The Ruler
Chapter 4 - Saving and Loading Text
```

Each line of text from the file TextAA is read into @0 by the macro. Of course other items could be included in the macro by reading them into other @ definitions. For example the fourth line of the macro definition could be extended to read in a page number as follows:

```
LJ Chapter 4C - @0 Page @1
```

And then the TextAA file can have the page numbers added so that we know on which page each chapter starts, thus:

```
AA The Wordprocessor,3
```

Macros can also contain highlight codes for selected sections of text. The chapter and number can be emboldened by inserting a highlight 2 code thus:

```
LJ Chapter 4C - @0 Page @1
```

Chapters, Sections and Sub-headings

The above example shows an application of a macro acting on a text file. Simple as it may be, it shows most of the principles involved and can be used as foundation for building up more complex macros. In the following example a series of macros are defined that will allow a document to be numbered according to chapter, section and sub-section as illustrated earlier in Chapter 10. Work through the example yourself - the first file is the macro definition file and as

before the comments in brackets are for information and should not be entered into the macro definition itself.

```

DM IN                                (define Initialisation macro IN)
DF ///0/                            (define footer)
DH ///1//                          (define header)
SR P 0                             (set page count to 0)
SR C 0                             (set chapter number to 0)
SR A 0                             (A = section number - set to 0)
SR B 0                             (B = subsection number - set to 0)
EM                                  (end macro definition)

DM CH                                (define macro CH for CHapter)
PE                                (eject page to start on new page)
SR C |C+1                          (increment chapter number by 1)
CE Chapter |C                      (print chapter details in centre)
CE 0                               (read text and print chapter title)
SR A 0                             (set/reset section count)
SR B 0                             (set/reset subsection count)
                                  (print blank line above section title)
EM                                  (end macro definition)

DM SE                                (define SEction macro SE)
LM 5                               (indent all sections by 5)
SR A |A+1                          (increment section count by 1)
SR B 0                             (set/reset subsection count)
LJ |C.|A. |A. 0                   (print chapter, section and title)
EM                                  (end macro definition)

DM SS                                (define sub-section macro SS)
LM 8                               (indent all sub-sections by 8)
SR B |B+1                          (increment sub-section count by 1)
LJ |C.|A.|B. |B. 0               (print chapter,section,sub-section & title)
EM                                  (end macro definition)

```

That completes the macro definitions, which will work for all documents that require numbering in the chapter-section-subsection manner. You can save this file to disc:

SAVE MacroCH

To use this macro all we need to do is to pass information into the correct macro as and when required. All documents must start with an

IN macro call to pass the footer and header details into the macro, then the CH macro should be used to provide the chapter title. After this SE macros can be used and these may contain as many SS macros as required. Type NEW and then enter the following text to work with macroCH (remember that the macro names must be entered into the stored command margin).

IN VIEW: A Dabhand Guide by Bruce Smith, Page 1P

CH The Wordprocessor

SE The Letter

SS A Sample Wordprocessed Letter

SS How The Wordprocessor is Used

SE Equipment

SS An Overview

SS Tape, Discs and Networks

SS Printer Choice

SE A Different VIEW

SS A Constant VIEW

SE Into VIEW

SS How To Enter VIEW

CH Introducing Command and Edit Mode

SE Command Mode

SS Status Information

SS Selecting MODE

SE The Status Information

SS Bytes Free

SS Editing

SS Screen Mode

SS Printer Driver

SE Edit Mode

This file can now be saved with a suitable name such as CONTENT

SAVE CONTENT

To produce a detailed contents list, all we now need do is load the macro file, macroCH followed by the file CONTENT:

```
PRINT macroCH CONTENT
```

alternatively we could preview it straight to the screen using the SCREEN command:

```
SCREEN macroCH CONTENT
```

Now each time VIEW encounters a line beginning with the stored command CH it will call upon the CH macro to print Chapter and the number followed by the chapter title defined in the CH macro call. Then the C register will be incremented by 1. The section and sub-section macros will be treated in the same way so that the result of printing the two files is as follows for the first chapter:

```
Chapter 1
The Wordprocessor

1.1 The Letter
1.1.1 A Sample Wordprocessed Letter
1.1.2 How The Wordprocessor Is Used

1.2 Equipment
1.2.1 An Overview
1.2.2 Tape, Discs and Networks
1.2.3 Printer Choice

1.3 A Different VIEW
1.3.1 A Constant VIEW

1.4 Into VIEW
1.4.1 How to enter VIEW
```

Subsequent chapters will be numbered accordingly. If we decide to alter the order of the documents we can simply perform a block move to re-arrange the contents in the CONTENTS file and reprint the lot again to obtain the new numbering. Macros are therefore very versatile, and well worth mastering if you do this kind of work.

13 : Long Documents

Appendix I contains details on ways of gaining more bytes free in VIEW, but even even with those measures there may well come a time where a document is simply simply too big for the memory. The obvious answer is to save the current text as part 1 of the document and carry on writing part 2, and indeed it is the technique I tend to use most of the time. This book is being written on a Master Compact and the majority of chapters can be saved as one file, but a few of the longer ones need to be broken down into two parts and saved accordingly, for instance as CHAP10a and CHAP10b. Sometimes however, breaking a document into smaller sections makes writing and editing it difficult. In these cases VIEW's continuous processing capability comes into its own.

The EDIT command allows VIEW to work on very long documents by reading in a part of the document from the big file under which it is saved. When you have finished work on this section you can move onto the next section which VIEW will read in, after of course saving the current section. To enable VIEW to do this it needs to maintain two files on your disc (it will not work on cassette). These are the input file, from which text is read, and the output file, to which text is saved. To start an EDIT session you use the command:

```
EDIT <FileIN> <FileOUT>
```

Demon Disc

A word of warning: before you start an EDIT operation it is very important that you have enough room on your disc for both the input and output files. If you don't and you get a 'Disc full' or 'Can't extend' error message then you're in acute trouble. To be safe, always use EDIT in conjunction with a disc that contains no other files. This does not necessarily apply if you are using ADFS, where typing

*FREE

from Command mode will show you how much space is left on the disc and if this is in excess of 100,000 bytes then you should be safe for all but the longest documents.

The Input File

Before you can start using EDIT, an input file must exist on the disc. This can be created simply by loading in your default ruler and headings from your Standard disc, swapping discs and saving the file under the input file name. The file name choice should include IN to distinguish it from the output file which should end with OUT. With the continuous processing disc in the drive and a small dummy file on the disc we can begin the EDIT with the command:

```
EDIT textIN textOUT
```

The disc will whirl somewhat longer than normal as VIEW creates the output file and reads in the text from the input file. At the end of it all the status information will be a line longer and will look somewhat like this:

```
Bytes free 26000  
Editing textIN to textOUT  
Input file is empty  
Screen mode 3
```

The Bytes free count will vary according to machine. The Editing line defines the name of the input and output files while the third line states that there is no more text held in the input file, ie textIN. When you are editing a very long document that cannot be all read into memory then the extra status line will tell you so:

```
Input file is not empty
```

Use of the command SAVE by itself will now give a Bad filename error, but SAVE can still be used to save the current contents of VIEW provided a filename is specified, ie,

```
SAVE Part1
```

The Output File

Text continues being entered into VIEW as usual, and, as in normal use, when memory is full the 'Memory Full' message will be displayed in inverse video over the ruler. Now we need to write the text currently in memory to the output file. This is done by moving into Command mode and typing:

```
MORE
```

The current text will be written to the output file and you will be left with a clean sheet from which to start. The default ruler will be restored so it will be necessary for you to reset this if so desired - there is no need to redo any other settings as they will be stored at

the start of the output file. Once again when memory is full MORE will save the current text onto the end of the text already in the output file and leave you with a blank sheet to continue with.

When you have completed the document then VIEW will need to transfer the remaining text to the output file and then close both the input and output file - this is done in Command mode with the command:

`FINISH`

Because VIEW is dealing with 2 files this will take a few seconds to accomplish so do be patient!

Now that all text has been written to the output file the input file is redundant and can be deleted:

`*DELETE textIN`

Should you wish to return to editing the text then the output file will become the input file and as such should be renamed:

`*RENAME textOUT textIN`

Marker 1 can be used to load text from the input file while saving some but not all to the output file. Set marker 1 at the point where you wish text to be transferred to the output file. Then in Command mode type:

`MORE 1`

All text above marker 1 will be transferred to the output file, with text being read from the input file to the bottom of the current text.

There is one final command associated with EDIT. The QUIT command will stop the editing process leaving the input file intact and without updating the output file. Needless to say this command should only be used at the very start of an Edit otherwise you will be left with two incomplete files.

Printing Long Documents

Although the SHEETS command can be used to print the pages of an EDIT document, loading in each section with MORE before continuing, it is not likely to remember previous stored commands. Long documents should then always be printed direct from disc with the `PRINT <filename>` command. To preview the whole file from disc at any time, use

`SCREEN <filename>.`

14 : Hints & Tips

Abbreviations

So far all of the commands used in Command mode have been entered in their entirety, but in fact the majority can be abbreviated to save time. Table 14.1 shows each command and the minimum abbreviation (NB. This only applies to VIEW versions 3.0 and above. See Appendix L for VIEW versions 2.1 and 1.4).

CHANGE	C
CLEAR	CL
COUNT	CO
EDIT	E
FINISH	F
FOLD	FD
FORMAT	FOR
LOAD	L
MICROSPACE	MI
MODE	M
MORE	MO (will perform MODE if no EDIT file active)
NAME	N
NEW	NEW
PRINT	P
PRINTER	PRINTE
QUIT	QUIT
READ	RE
REPLACE	R
SAVE	SA
SCREEN	SC
SEARCH	S
SETUP	SET
SHEETS	SH
WRITE	W

Table 14.1 Minimum Command Abbreviations.

When using any of the above abbreviations always insert a space between the abbreviation and any parameter otherwise an error will occur ie, it's:

```
!M 3
```

and not

```
!M3
```

Star Commands

A variety of *TV and *FX commands can be used from Command mode or included in your !Boot file. Here are some of the more interesting ones.

*TV 0,1 : Text on the screen can sometimes seem to judder or shimmer - this command will stop it. Note that it must be followed by a MODE change to take effect, ie,

```
*TU 0,1
MODE 3
```

*TV 255,1 : On some monitors the text can be very close to the top of the screen, and this command will move the screen display down by 1 line. This can be set by a *CONFIGURE command on Master 128 and Master Compact. The default is normally *TV255,1.

*FX 5 : This is used to select whether printer output should go to the serial or parallel port. *FX5,1 will select the parallel port for use with a standard Centronics printer of the Epson type. Typing *FX5,2 will select a serial printer and all printer output will be directed to the RS serial port. *FX5 on its own disables printer output.

*FX 6 : This command has a default of *FX6,10 and as such causes linefeeds to the printer to be ignored. If you have printer dip switches set to send linefeeds then you need not use this command. If on the other hand linefeeds are not enabled via the printer dip switches, then *FX6,0 should be entered either from Command mode or as part of your standard !Boot file.

*FX 8 : Is used in conjunction with a serial printer to select the transmission rate, that is, the speed at which data is sent to the printer. On a Master 128 or Master Compact these can be 'permanently' set with the *CONFIGURE BAUD and *CONFIGURE DATA commands. The serial settings are as follows:

*FX 8,1	75 Baud
*FX 8,2	150 Baud
*FX 8,3	300 Baud
*FX 8,4	1200 Baud
*FX 8,5	2400 Baud
*FX 8,6	4800 Baud
*FX 8,7	9600 Baud
*FX 8,8	19200 Baud

*FX11 : Hold down any key on the keyboard and it will 'repeat' after a short delay. This delay may be set with this command: *FX 11,n where n is a number which is the delay in 100ths of a second.

*FX12 : While *FX11 allows you set to set the delay repeat rate, this command allows you to set the rate at which the key will be repeated. The default is *FX12,8 and the second parameter if not zero sets the delay between successive characters being printed in 100ths of a second.

*FX202 : The command *FX202,48 will turn the CAPS LOCK light off and set the keyboard to lower case characters - this is a useful item to include in a !Boot file.

*FX228 : Using *FX 228,1 allows function keys to be used in Edit mode. Keys can be programmed with character strings which can then be inserted into the text by holding down the function key with CTRL and SHIFT. If function key 0 was programmed from Command mode with:

*KEY0 VIEW : A Dabhand Guide

and *FX228,1 had been previously issued, then pressing CTRL-SHIFT-10 (all three keys together) when in Edit mode would insert 'VIEW : A Dabhand Guide' into your text. See Highlighting Problems below for further examples.

A Touch of Colour

If you are using a colour monitor or TV for your wordprocessing and are working in a screen mode other than Mode 7 (and Mode 135 on a Master or B+) then it is possible to change the colours of both the background (paper) and text (ink). There are 8 colours in all and these are assigned numbers as follows:

0 = black	4 = blue
1 = red	5 = magenta
2 = green	6 = cyan
3 = yellow	7 = white

By default the background is black and the text is white, ie, the default colours are 0 and 7. We can choose each of these if so desired by entering Command mode and then pressing CTRL-S then the number of the colour (always 0 for background or 7 for text), the new colour number to be assigned to it and then 3 zero's. For example, to obtain green text on a black background we need to change the text colour 7 to green, which is 2, so we press:

```
CTRL-S 7 2 0 0 0
```

with no spaces or RETURNS. Note that the numbers will not be printed onto the screen. To reset to white text we would use:

```
CTRL-S 7 7 0 0 0
```

To change the background colour then the first number specified must be 0, so to set a blue background use:

```
CTRL-S 0 4 0 0 0
```

where 4 is the colour blue. If you get into a mess setting colours, and you can no longer see what you're typing, then press CTRL-T a few times, and the display will return to black and white.

!Boot Files

!Boot files can be customised to your own requirements and these can be written directly in VIEW. Simply enter VIEW in the normal way, type NEW and then in Edit mode write your file, remembering to keep instructions to one per line. A small !Boot file written in VIEW might contain the following (the text in brackets should not be entered, it is for information only):

*WORD	(Select VIEW)
*TV 255,1	(Move screen down one line and cure judder)
MODE 3	(Select Mode 3 or 131 on Master)
NEW	(New text - reset ruler)
SETUP FJ1	(Select text formatting required)
LOAD standard	(Load standard page header details)
NAME	(Clear NAME setting)
PRINTER FX80	(Load printer driver)
*EXEC U.Extkeys	(Load function keys - see below)

Once the !Boot file is complete it should be saved to disc using the WRITE command:

```
WRITE !Boot
```


The disc boot option should then be set to *EXEC by typing:

*OPT 4,3

If you wish to edit the !Boot file it can be READ back and edited as required.

Highlighting Problems

The Acornsoft Printer Driver Generator allows various printing effects to be produced. At the simpler level emboldening and underlining are available, but by switching to the extended highlight sequence a wider range can be obtained.

Listing 14.1 is a way of defining the function keys to allow each of the extended highlights to be 'typed' with a single keypress.

```
*| V.Extkeys quick entry of extended highlights  
*| by Graham Bell  
*fx 228 1  
*fx 1B  
*key 0  
*key 1  
*key 2 "|||| |!"  
*key 3 "|||! "  
*key 4 "| |||!!"  
*key 5 "||||!"  
*key 6 "|||! |!"  
*key 7 "| |||!!!"  
*key 8 "! ! |!"  
*key 9 "| |$HT|M2 130|M|!!!! |! |!"  
*key 10  
*key 11 "| |@"  
*key 12 "| |X| |X| |X| |X| |X| |X| |X| |X| |X|"  
*key 13 "| |Y| |Y| |Y| |Y| |Y| |Y| |Y| |Y| |Y|"  
*key 14 "| |Z| |Z| |Z| |Z| |Z| |Z| |Z| |Z| |Z|"  
*key 15 "
```

Listing 14.1. The Extkeys listing.

The listing is not a program: it can be typed in with VIEW itself, then saved as a file as follows:

WRITE V.Extkeys

This could then be added to your !Boot file or simply executed from Command mode. In both cases the command is:

*EXEC U.Extkeys

The function keys are defined to give the following extended highlight sequences:

SHIFT-CTRL-I2	alternative (Elite) font on-off
SHIFT-CTRL-I3	subscript on
SHIFT-CTRL-I4	sub- or superscript off
SHIFT-CTRL-I5	superscript on
SHIFT-CTRL-I6	italics on-off
SHIFT-CTRL-I7	bold on-off
SHIFT-CTRL-I8	index mark (for ViewIndex)
SHIFT-CTRL-I9	reset printer
SHIFT-CTRL-COPY	exact space (pad character *@)

Furthermore, the cursor keys with SHIFT-CTRL will move the cursor eight times faster than the cursor keys alone. The keys have to be pressed in combination with both shift and control together, so as not to clash with VIEW's built-in editing functions. In listing 14.1, the function keys f0 and f1 are left free for your own definitions.

Of course, all these effects can be put on a function key strip to remind you of their action.

DIY Editing

There are some functions that are conspicuous by their absence from VIEW. One of these is a delete word key. To delete a word 'by hand', the cursor is moved to the first letter of the word, then SHIFT-I3 DELETE UP TO CHARACTER is pressed, then the space bar. This only works on VIEW 3. It can be programmed into a function key as follows:

```
*key 0 "|||_ "
```

Note that there is a space before the final quote mark. So when the cursor is on the first letter of a word, pressing SHIFT-CTRL-I0 will delete the word in one go. The "|||_" represents the SHIFT-I3, and it is followed by a space. An improvement upon this definition is:

```
*key 0 "|||Y|||Y|(|_|_ "
```

because the cursor doesn't need to be under the first letter of the word to be deleted (anywhere from the space immediately preceding the word to the character before the space preceding the next word will do). Unfortunately, neither definition will delete the last word on a line.

Note the use of quotation marks: they are not normally needed for defining soft key strings, but they do make spaces at the beginning or end of the string visible (there is one space at the end of each of the above definitions). If typed in without quote marks, trailing spaces are taken to be part of the string but leading spaces are ignored: if necessary, a leading space can always be specified with ' '.

Almost any series of keypresses can be programmed into a red key. Table 14.1 lists the strings that VIEW itself uses for each key. Using these codes you can construct your own key functions which call the built-in functions. The only two points to watch are that ESCAPE can only be emulated in text mode (so '||' in a *key definition will make the wordprocessor return to command mode, but not go from command mode to edit mode), and that one soft key definition may not refer to another (you can't incorporate the delete word function directly into another definition, so there is no string given for SHIFT-CTRL-10).

Using this soft key string technique, numerous useful macros can be invented. For example, a delete to beginning of line function might be needed:

```
*key 1 "||*2||P||*|||,"
```

If, after looking at Table 14.1, the workings of this and the improved delete word definition are clear, then writing new functions should be no problem. Otherwise, think of the definitions split up into steps as shown in Figure 14.2. The following command is also included in the V.Extkeys file:

```
*fx 228 1
```

This must be done in command mode, to ensure that SHIFT-CTRL-function key combinations are interpreted as soft keys otherwise they will not work! *FX18 in the V.ExtKeys file clears any old definitions prior to redefining them.

f0	L	SHIFT-f0	\	CTRL-f0	,
f1	H	SHIFT-f1]	CTRL-f1	-
f2	N	SHIFT-f2	^	CTRL-f2	.
f3	O	SHIFT-f3	_	CTRL-f3	/
f4	P	SHIFT-f4		CTRL-f4	0
f5	Q	SHIFT-f5	!	CTRL-f5	1
f6	R	SHIFT-f6	"	CTRL-f6	2
f7	S	SHIFT-f7	*	CTRL-f7	3
f8	T	SHIFT-f8	\$	CTRL-f8	4
f9	U	SHIFT-f9	x		

NB: SHIFT-f4 includes a single space after the !

Cursor keys:

X	left	(SHIFT-left
Y	right)	SHIFT-right
Z	down	*	SHIFT-down
[up	*	SHIFT-up

8	CTRL-left
9	CTRL-right
;	CTRL-down
;	CTRL-up

Other special keys:

	TAB	W	COPY
?	DELETE	M	RETURN
'	SHIFT-COPY	[ESCAPE

NB: To insert " or | into a key string, use |" or ||

Table 14.1 - Function key codes used by VIEW.

*	SHIFT-f7	SET MARKER
2		Marker number two
P	f4	BEGINNING OF LINE
*	SHIFT-f7	SET MARKER
1		Marker number one
,	CTRL-f0	DELETE BLOCK between markers

Table 14.2 - Delete to beginning of line.

V	right	two spaces right
V	right	
(SHIFT-left	go back to start of word
_	SHIFT-f3	DELETE UP TO CHARACTER
SPACE		delete up to next space

Table 14.3. Improved Delete Word.

*CONFIGURE

Master 128 and Master Compact owners will find that *TV255,1 is already set as part of the micro's *CONFIGURE system. Typing:

*STATUS

will provide a list of settings. The Master Compact also allows VIEW's SETUP command to be included in the *CONFIGURE sequence. Typing *STATUS will show that the setting is:

SETUP FI

If you wish to set this to include justification as well then enter the following command from Command mode:

*CONFIGURE SETUP FJI

Pounds and Hashes

In Chapter 11 we saw why it is seemingly impossible for pounds and hashes to be present on the printer together - but isn't! The only requirement is that you have a Epson FX80, Kaga Taxan or Canon printer or any other printer which has downloadable character fonts. If you are in doubt then try the short program below and see if it works by enabling the printer and typing pounds and hashes.

The program is very short but must be run in BASIC so if you plan to use it save it using a suitable filename (ie, £hash) and make your !Boot file chain this ie,

```
CHAIN "£hash"
```

You can include an extra line at the end of the program to execute your normal boot file which should have been given a new name.

```
10      VDU 2
20      VDU 1,27,1,ASC"R",1,0
30      VDU 1,27,1,ASC":",1,0,1,0,1,0
40      VDU 1,27,1,ASC"%",1,1,1,0
50      VDU 1,27,1,ASC"&",1,0,1,96,1,96
60      VDUI1,138,1,18,1,0,1,126,1,128,1,18,1,128
      ,1,18,1,128,1,66,1,0,1,0
70      VDU 3
80      REM *EXEC !new
```

15 : VIEW Manager

Presented in this chapter are two programs which can be used either individually or together to form what I have called the VIEW Manager system. VIEW Manager offers a visually pleasing 'front end' to VIEW which can be included as the main !Boot option of your VIEW disc and allows selection via a menu of:

- * Load - easy loading of VIEW files
- * ExCat - an extended catalogue system
- * Template - easy selection of dummy template files
- * Precis - a brief description of files on disc

ExCat

The Extended Catalogue program will look at each of your VIEW files in turn and for any that contain a special comment line at the start, will read it and display the comment. This allows you a description of 40 characters which may take any form. Each file is numbered, and selecting the number will load that file into VIEW ready for editing. Using this form of cataloguing means that filenames can be more ambiguous.

The ExCat program (Listing 15.1) can be found at the end of this chapter, and of course on the Program Disc (see Appendix M). The program is written entirely in BASIC and should present you with no problems to enter. When you have entered and checked the program, save it onto your VIEW disc using the filename ExCat:

```
SAVE "ExCat"
```

For ExCat to work as intended, each of your VIEW files must have the special comment line right at the top - even before any ruler. Existing files can be modified simply by inserting a new blank line at the top of the document using function key f6 INSERT LINE. The comment line begins with a stored command, CO, which should be inserted into the stored command margin. The comment can then be typed on the same line in the edit area, as in this example:

CO VIEW: A Dabhand Guide - Chap15 - VIEW Manager

There is no limit to the number of commented VIEW files other than the restriction of files in your filing system. If a VIEW file does not contain a CO line then it will have no adverse effect, and similarly BASIC programs or any other type of file will simply be ignored.

There is only one change that you will need to make to the ExCat program. This is in line 790, where N\$ must be set to the directory in which you keep your VIEW files. By default it is set to VIEW, but if you are using DFS then the directory name will be limited to a single character: if for example you use directory V for VIEW files then this will be V. If you don't use the directory system then simply set N\$="\$". ADFS users should use the complete directory root, although the \$ need not be specified.

Only 15 files are displayed on screen at one time - should the number of CO files exceed this then only the first 15 will be displayed - pressing a key will show the remaining files, at which time the appropriate file number can be entered. The total number of files in the directory, regardless of whether they contain the CO line, is also displayed.

Figure 15.1 shows the display provided by ExCat when run on my own system. If I wished to load Chapter 13 of the book I would simply enter 2 and press RETURN.

View Manager Extended Catalogue

22 Files

1	CH14	VIEW: A Dabhand Guide	Chapter 14	Hints and Tips	10
2	CH13	VIEW: A Dabhand Guide	Chapter 13	Long Documents	16
3	CH12	VIEW: A Dabhand Guide	Chapter 12	Macros Revisited	13
4	CH11	VIEW: A Dabhand Guide	Chapter 11	Printers and Drivers	11
5	CH10	VIEW: A Dabhand Guide	Chapter 10	Macros	4
6	CH9	VIEW: A Dabhand Guide	Chapter 9	Page Layout	16
7	CH8	VIEW: A Dabhand Guide	Chapter 8	Stored Commands and Margins	10
8	CH7	VIEW: A Dabhand Guide	Chapter 7	Search and Replace	17
9	CH6	VIEW: A Dabhand Guide	Chapter 6	Formatting and Justification	16
10	CH5	VIEW: A Dabhand Guide	Chapter 5	Markers	17
11	CH4	VIEW: A Dabhand Guide	Chapter 4	Saving and Loading Text	20
12	CH3	VIEW: A Dabhand Guide	Chapter 3	The Rules	13
13	CH2	VIEW: A Dabhand Guide	Chapter 2	Command and Edit Mode	5
14	CH1	VIEW: A Dabhand Guide	Chapter 1	The Wordprocessor	4

Which numbered file

Figure 15.1 - ExCat display.

VIEW Manager

The VIEW Manager program is Listing 15.2 and this makes use of the ExCat program as one of its options. The program is written mainly in BASIC although there is a small section of assembler - again this should present no problems. Until the program is running correctly you should not enter line 60. The program is also on the Program Disc.

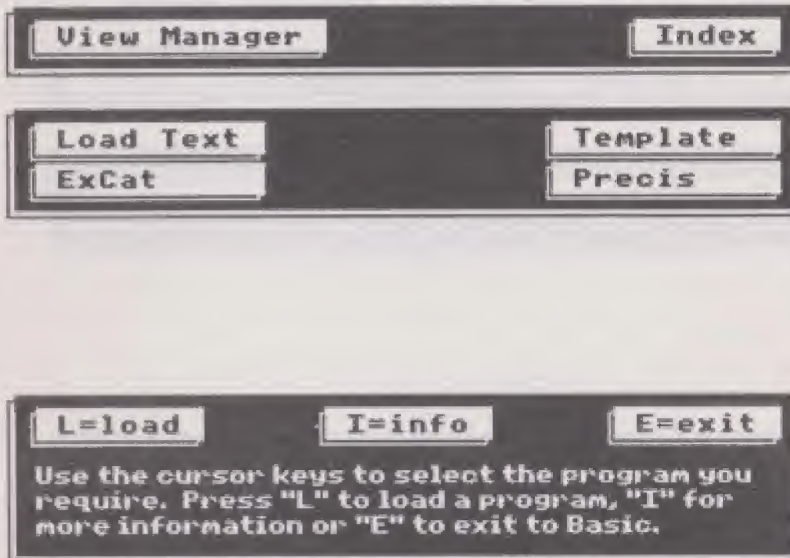


Figure 15.2. The VIEW Manager front end.

Figure 15.2 shows the VIEW Manager front end - the four cursor keys can be used to move around the options. The currently selected option will appear inversed. Pressing L will load that option; pressing I will provide a small amount of information about the currently selected option and pressing E will leave the VIEW Manager system and go into to BASIC.

What follows is a description of each of the options provided by the program and how to use them.

Load Text

This option allows you to enter VIEW and load a file of your choice. After selecting Load Text, you will be prompted to enter a filename. This should be typed, including the full directory root if you are using ADFS, and RETURN pressed. VIEW will be entered and the file loaded. If the file does not exist then you will remain in VIEW with an error. Figure 15.3 shows the screen when Load is selected.

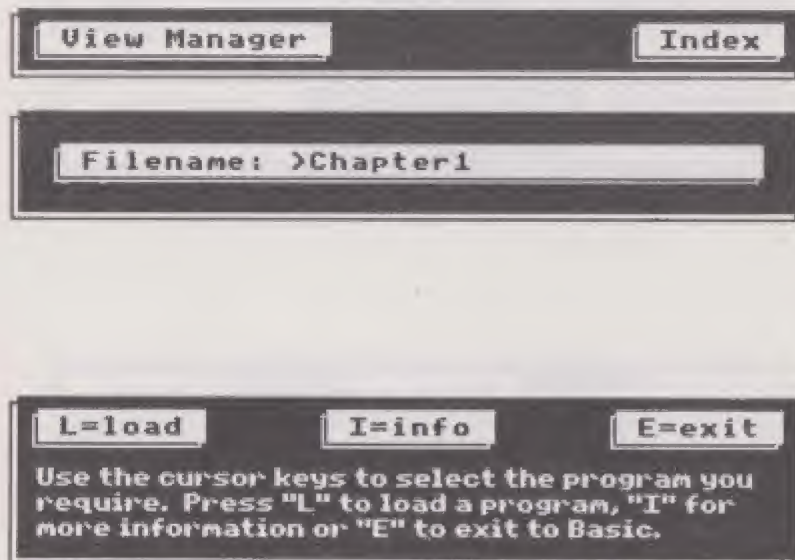


Figure 15.3. The load screen.

ExCat

This option will simply chain the EXCAT program. This should therefore be included on the same disc as VIEW Manager.

Template

Figure 15.4 shows the Template Index screen. The templates are VIEW files containing standard start files. Examples include Letters, Memos, Chapters and so forth. To add your own template details you

will need to edit the VIEW Manager listing. As listed it contains two choices in lines 3200 to 3220. Each entry is placed in a DATA line and takes the following form:

```
DATA <item>,<filename>,<description>
```

The <item> will be displayed on the screen and used to make a selection from - it needs to be concise but descriptive and no longer than 10 characters. The <filename> is the name of the template file as stored on the disc. The <description> is what will be displayed on pressing the I (info) option from the menu screen.

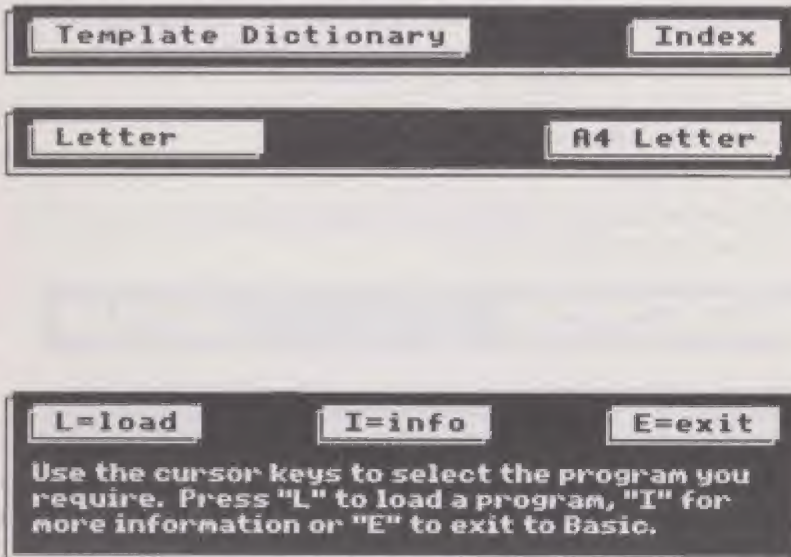


Figure 15.4. The template index screen.

Precis

This option allows you to read a special VIEW file which contains a precis of the files on the disc. Each precis may be up to 255 characters long and so offers a more detailed description of a file than that given by ExCat.

The PRECIS file should be saved on the same disc as VIEW Manager but must conform to a certain style as follows:

```
@<Filename>
Text
@<Filename>
Text
```

The @ character is used to prefix the name of the file on the disc - this should be followed, on the next line(s) by a description of the file, which should not exceed 255 characters. This can then be repeated; figure 15.5 shows a typical PRECIS file.

```

VIEW: A Dohdard Guide
An introduction to VIEW, ViewSpell & ViewIndex by Bruce Smith. Published
by Doba Press.

```

Figure 15.5. A PRECIS file.

Once complete it should be written to the disc:

WRITE PRECIS

The file can be loaded, added to and edited at any time. Figure 15.6 shows the Precis option in action.

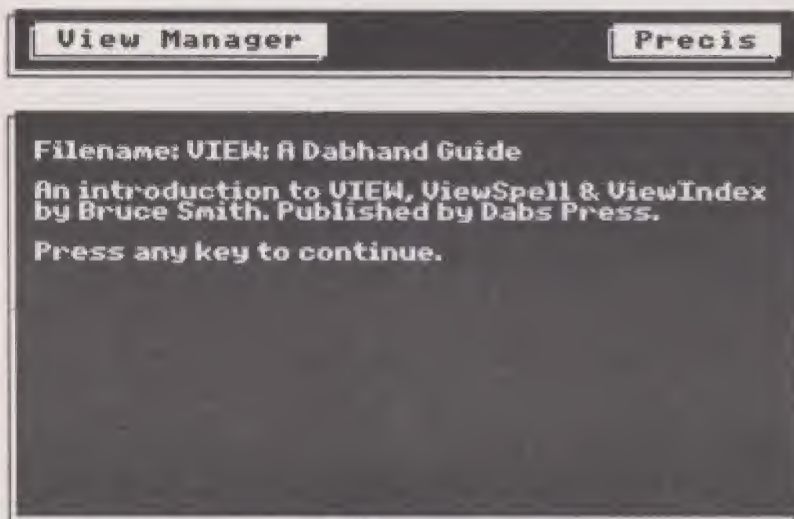


Figure 15.6. A PRECIS file in action.

Extending VIEW Manager

VIEW Manager can be extended in a number of ways. The first is when loading files using either the Load or Template options - you may wish to include your own !Boot files to set up VIEW ready for use on the file just loaded. This can be done by editing line 240 (for Load) and/or line 2830 (Template). VIEW Manager itself should be included as the main !Boot file by as follows:

```
*BASIC
CHAIN "Vmanager"
```

Where Vmanager is the filename used to save the VIEW Manager program. We can include a !Boot file but must use a different name, and be sure that the new file does not include commands such as *WORD or NEW otherwise the text just loaded will be erased!

Let us suppose we have written a new !Boot file and called this !Lboot (for Load boot). This should be *EXECed and line 240 edited thus:

```
240 PROCoscli("KEY9LOAD "+!f$+"|N*EXEC
!Lboot|N*KEY9|N")
```

You may also wish to include options to enter VIEWSpell, VIEWStore and VIEWSheet if these are installed in your machine. These and any other options can be added as follows. First add the details in the form of DATA statements between lines 3160 and 3170 of the VIEW Manager program. The DATA statement must take the form:

```
DATA <item>,<filename>,<description>
```

which is the same as for adding templates (see Template above). The <filename> is that of a BASIC program that will select the option and execute any boot files. To include VIEWSpell, VIEWSheet and VIEWStore the following DATA lines could be used:

```
3161 DATA "ViewSpell","VSPELL","Load ViewSpell"
3162 DATA "ViewSheet","VSHEET","Load ViewSheet"
3163 DATA "ViewStore","VSTORE","Load ViewStore"
```

Here VIEW Manager would expect to find BASIC programs called VSPELL, VSHEET and VSTORE on the disc. These can then be run and should select the ROM in question and perhaps execute a boot file of sorts.

Listings

Listing 15.1.

```

10 REM Extended Catalogue
20 REM for use with CO stored command
30 REM (C) Bruce Smith 1987
40 REM VIEW: A Dabhand Guide
50 :
60 ON ERROR GOTO 980
70 MODE 3
80 HIMEM=62E00
90 PROCsetup
100 PROCreadcat
110 PROCshowfiles
120 END
130 :
140 DEF PROCsetup
150 DIM F$(50), buf% 100
160 DIM readcat $50, dir% 50
170 bkt%=$A00:str%=$A40
180 oac1%=$FFF7
190 oaword=$FFF1
200 *FX 18
210 *FX 12
220 ENDPROC
230 :
240 DEF PROCshowfiles
250 CLS
260 PRINT "VIEW Manager Extended Catalogue"
270 PRINT num%+1;" Files"
280 PRINT STRING$(79," ")
290 @% = 2: T% = 0: size% = 0
300 PRINT
310 FOR Z% = 0 TO num%
320 IF dir% Z% = 2 THEN GOTO 380
330 N% = OPENUP F$(Z%)
340 size% = size% + EXT#N%
350 F% = BGET#N%
360 IF F% = 128 THEN PROCprint
370 CLOSE#N%
380 NEXT
390 PRINT STRING$(79," ")
400 INPUT "Which numbered file " file%
410 IF file% > T% THEN VDU7,11:PRINTSPC(30):VDU11:GOTO 400
420 IF file% = 0 THEN RUN
430 N% = EXT#OPENUP F$(file%-1)
440 CLOSE#0
450 IF N% > 9500 THEN MODE 7
460 PROCoscll("KEY9LOAD "+F$(file%-1)+"|M*KEY9|M")
470 PROCview

```

```

480 END
490 :
500 DEF PROCprint
510 F$(T%)=F$(Z%)
520 PRINT T%+1 TAB(3)F$(T%)TAB(13);
530 PTR#N%=PTR#N%+2
540 FOR C%=1 TO 63
550 F%=BGET#N%
560 IF F%=13 THEN C%=64 ELSE PRINT CHR$(F%);
570 NEXT
580 PRINTTAB(??){EXT#N%+500}DIV 1000
590 T%=T%+1
600 IF T%=15 THEN VDU7:PRINT".. ANY key to CONTINUE
..":IF GET THEN VDUI1:PRINTSPC(27):VDUI1,11
610 ENDPROC
620 :
630 DEF PROCview
640 *FX 138,0,137
650 *FX 12,6
660 *FX 11,15
670 *WORD
680 ENDPROC
690 :
700 DEF PROCoscli(ccline$)
710 $buf%=ccline$
720 X%=buf%
730 Y%=buf% DIV 256
740 CALL oscli%
750 ENDPROC
760 :
770 DEF PROCreadcat
780 *DIR
790 N$=" $"
800 PROCoscli("DIR "+N$)
810 ?bk%=0:bk%+1=HIMEM
820 bk%+5=50:bk%+9=0
830 X%=0:Y%=&A:A%=8:CALL&FFD1
840 num%=49-bk%+5:L%=?HIMEM+1
850 FOR I%=0 TO num%
860 F$(I%)=""
870 FOR J%=1 TO L%-1
880 F$(I%)=F$(I%)+CHR$(?(HIMEM+I%*L%+J%))
890 NEXT
900 $str%=F$(I%):!bk%=str%
910 X%=0:Y%=&A:A%=5
920 F%=USR(&FFD0) AND 255
930 dir%?I%=F%
940 NEXT
950 ENDPROC
960 :
970 REM Error Handler

```

VIEW : A Dabhand Guide

```
980 MODE 7
990 PRINT""Error""
1000 REPORT
1010 PRINT" at line ";ERL
1020 PRINT"Press any key to continue";
1030 key=GET
1040 RUN
```

Listing 15.1 - Extended catalogue.

Listing 15.2.

```
10 REM VIEW Manager
20 REM needs EXCAT and PRECIS file
30 REM (C) Bruce Smith 1987
40 REM VIEW: A Dabhand Guide
50 :
60 ON ERROR GOTO 3020
70 RESTORE
80 MODE4:VDU 6,23,1,0;0;0;0;
90 PROCass
100 DIM p$(12),f$(12),i$(12),buf% 100
110 *FX17B,255
120 *FX229
130 PROCdoscreen
140 PROCselect
150 IFk%=101 MODE 6:END
160 PROCinv(ch%)
170 IF f$(ch%)="PRECIS" THEN PROCprecis:END
180 IF f$(ch%)="TEMP" THEN PROCtemplate:RUN
190 IF f$(ch%)="ExCat" THEN CHAIN"ExCat"
200 PROCw(1,6+2*h%,38,6)
210 PROCbox(3,8,36):INPUT" Filename: >"lf$
230 MODE 7:REM NB there is no line 220
240 PROCoscli("KEY9LOAD "+lf$+"|M*KEY9|M")
250 PROCview
260 END
270 :
280 DEF PROCinfo
290 PROCcls:PROCinv(ch%)
300 PROCw(1,30,38,20)
310 PROCwcp(21,15(ch%)+ " Press the RETURN key to
continue.")
320 *FX15,1
330 REPEAT UNTIL GET=13
340 PROCinv(ch%)
350 ENDPROC
360 :
370 DEF PROCalter(a%)
380 PROCinv(ch%):ch%=ch%+a%
390 IF ch%>=n% ch%=n%-1
```



```

400 IF ch%<0ch%=0
410 PROCinv(ch%)
420 ENDPROC
430 :
440 DEF PROCcls
450 COLOUR128
460 VDU28,0,31,39,20,12
470 ENDPROC
480 :
490 DEF PROCkeys
500 PROCcls
510 PROCw(1,27,38,20)
520 PROCi(2,"L=load")
530 PROCi(16,"I=info")
540 PROCi(30,"E=exit")
550 PROCw(23,"Use the cursor keys to select the program
you require. Press "L" to load a program, "I" for
more information or "E" to exit to Basic.")
560 ENDPROC
570 :
580 DEF PROCw(l,d,r,u)
590 VDU24,1*32-12;(31-d)*32-12;(r+1)*32-12;(32-u)*32-12;
600 GCOLD,131:CLG
610 VDU24,1*32-8;(31-d)*32-8;(r+1)*32-16;(32-u)*32-16;
620 GCOLD,128:CLG
630 COLOUR0:COLOUR129
640 VDU 28,1,d,r,u,12
650 ENDPROC
660 :
670 DEF PROCbox(l,d,r)
680 VDU24,1*32-8;(31-d)*32-16;(r+1)*32-8;(32-d)*32;
690 GCOLD,128:CLG
700 VDU24,1*32-4;(31-d)*32-12;(r+1)*32-12;(32-d)*32-4;
710 GCOLD,129:CLG
720 VDU24,1*32;(31-d)*32-8;(r+1)*32;(32-d)*32+8;
730 GCOLD,128:CLG
740 COLOUR128:COLOUR1
750 VDU28,1,d,r,d,12
760 ENDPROC
770 :
780 DEF PROCshow(p%) 790 IF p%<h% PROCbox(2,1+2*p%,3+m%)
ELSE PROCbox(36-m%,7+2*(p%-h%),37)
800 PRINT" " "p$(p%);
810 ENDPROC
820 :
830 DEF PROCi(c,c6)
840 PROCbox(c,21,c+7)
850 PRINT" "c$;
860 ENDPROC
870 :
880 DEF PROCinv(p%)

```

VIEW : A Dabhand Guide

```

890 GCOL4,128
900 IF p%<h%:VDU 24,68:32*(24-2*p%)-4;32*(m%+4)-4;32*(25-
2*p%)+4; ELSE VDU 24,1284-32*(m%+4);32*(24-2*(p%-h%))-
4;1212:32*(25-2*(p%-h%))+4;
910 CLG
920 ENDPROC
930 :
940 DEF PROCwrip(y%,t%)
950 VDU26:GCOL0,0
960 y%=1007-32*y%
970 REPEAT
980 IF ASCt%-32t%=-MID$(t%,1):GOTO980
990 IF LENT%<48 GOTO 1080
1000 n%=-LEFT$(t%,48):i%=49
1010 REPEAT:i%=i%-1
1020 UNTIL i%=1 OR MID$(n%,i%,1)="-"
1030 IF i%=1 i%=44
1040 n%=-LEFT$(t%,i%-1)
1050 t%=RIGHT$(t%,LENT%-i%)
1060 MOVE 60,y%
1070 PROCprop(n%):y%=y%-40
1080 UNTIL LENT%<48
1090 IF t%="" ENDPROC
1100 MOVE60,y%
1110 PROCprop(t%)
1120 ENDPROC
1130 :
1140 DEF PROCdoscreen
1150 READtt%
1160 n%=-1:m%=0
1170 REPEAT
1180 n%=n%+1:READ n%
1190 IF n%=""GOTO1230
1200 p%(n%)=n%
1210 READf%(n%),is(n%)
1220 IF LENn%>m% m%=LENn%
1230 UNTILn%=""
1240 h%=(n%+1)/2
1250 PROCw(1,3,38,1)
1260 PROCbox(2,2,3+LENT%)
1270 PRINT" *t%";
1280 PROCbox(31,2,37):PRINT" Index";
1290 PROCw(1,6+2*h%,38,6)
1300 FOR x%=0 TO n%-1
1310 PROCshow(x%):NEXT
1320 ch%=0
1330 ENDPROC
1340 :
1350 DEF PROCselect
1360 *FX4,1
1370 REPEAT

```

```

1380 PROCkeys:*FX15,1
1390 PROCinv(ch%)
1400 REPEAT:k%=GET
1410 IF k%<136 k%=k%OR32
1420 IF k%=136 AND ch%=>h% PROCalter(-h%)
1430 IF k%=137 AND ch%<h% PROCalter(h%)
1440 IF k%=139 AND ch%>0 PROCalter(-1)
1450 IF k%=138 AND ch%<n%-1 PROCalter(1)
1460 UNTIL k%=108 OR k%=101 OR k%=105
1470 PROCinv(ch%)
1480 IF k%=105 PROCinfo
1490 UNTIL k%=108 OR k%=101
1500 *FX4
1510 ENDPROC
1520 :
1530 DEF PROCass
1540 DIM code%170,ms%255
1550 char=470:count=471
1560 block%=473:work%=470
1570 FOR pass%=0 TO 2 STEP 2
1580 P%=code%
1590 IOP7 pass%
1600 .prop
1610 LDA #0:STA char
1620 .charloop
1630 LDX char:LDA ms%,X
1640 CMP #13:BNE notlastchar
1650 RTS
1660 .notlastchar
1670 CMP #32:BNE notspace
1680 LDA #12:STA pra+2
1690 LDA #0:$7A pra+3
1700 JSR doplot:BEQ nextchar
1710 .notspace
1720 STA block%
1730 LDX #block% MOD 256
1740 LDY #block% DIV 256
1750 LDA #5A:JSR $FF81:JSR copy
1760 .leftloop1
1770 LDX #7
1780 .leftloop2
1790 ASL work%,X:$CS leftbitfound
1800 DEX:BPL leftloop2:INY
1810 CPY #8:BNE leftloop1
1820 .leftbitfound
1830 CPY #0:BNE nottopwide
1840 STY pra+3:LDA #4
1850 STA pra+2:JSR doplot
1860 BEQ noleft
1870 .nottopwide
1880 DEY:JSR sub

```


VIEW : A Dabhand Guide

```

1890 .noleft
1900 LDA block%,JSR &FFEE:JSR copy
1910 .rightloop1
1920 LDX#7
1930 .rightloop2
1940 LSR work%,X:BCC rightbitfound
1950 DEX:BPL rightloop2:INY
1960 CPY #8:BNE rightloop1
1970 .rightbitfound
1980 JSR sub
1990 .nextchar
2000 INC char:BNE charloop
2010 .copy
2020 LDY #8
2030 .copyloop
2040 LDA block%,Y:STA work%-1,Y
2050 DEY:BNE copyloop:RTS
2060 .sub
2070 TYA:ASL A:ASL A
2080 BEQ noplot:STA count
2090 LDA #0:SBC SBC count
2100 STA prs+2:LDA #0:SBC #0
2110 STA prs+3
2120 .doplot
2130 LDX #0
2140 .doplotloop
2150 LDA prs,X:JSR &FFEE
2160 INX:CPY #6:BNE doplotloop
2170 .noplot
2180 RTS
2190 .prs
2200 }
2210 7P%=25:P%?1=0:P%?2=0
2220 NEXT pass%
2230 ENDPROC
2240 :
2250 DEFPROCprop($ms%)
2260 VDU5:CALL prop
2270 VDU4,23,1:0:0:0:0;
2280 ENDPROC
2290 :
2300 DEF PROCprecis
2310 CLS
2320 2%=OPENUP("PRECIS")
2330 REPEAT
2340 char%=BGET#2%
2350 UNTIL char%=64
2360 string$="":len%=0
2370 PROCw(1,3,38,1)
2380 PROCbox(2,2,3+LENt$):PRINT"  t:5;
2390 PROCbox(30,2,37):PRINT"  precis";

```

```

2400 PROCw(1,25,38,6)
2410 REPEAT
2420 char%=BGET#2%
2430 string$=string$+CHR$(char%)
2440 UNTIL char%=13
2450 string$="Filename: "+string$
2460 PROCw(7,string$)
2470 string$="":text%=0
2480 REPEAT
2490 text%=text%+1
2500 char%=BGET#2%
2510 IF EOF#2% THEN GOTO 2550
2520 IF char%=64 THEN GOTO 2550
2530 IF char%=&1A OR char%=13 char%=32
2540 string$=string$+CHR$(char%)
2550 UNTIL char%=64 OR EOF#2%
2560 :
2570 sp%=8:ct%=0
2580 REPEAT
2590 sp%=sp%+1
2600 part$="":len%=0
2610 REPEAT
2620 len%=len%+1:ct%=ct%+1
2630 char%=MID$(string$,ct%,1)
2640 part$=part$+char$
2650 UNTIL len%=47
2660 PROCw(1,sp%,part$)
2670 UNTIL ct%>(text%-1)
2680 :
2690 PROCw(1,sp%+2,,"Press any key to continue.")
2700 key%=GET
2710 IF NOT EOF#2% GOTO 2360
2720 CLOSE#0
2730 RUN
2740 END
2750 :
2760 DEF PROCtemplate
2770 RESTORE 3190
2780 CLS
2790 PROCdoscreen
2800 PROCselect
2810 IF k%=101 ENDPROC
2820 PROCinv(ch%)
2830 PROCoscli("KEY9LOAD "+fs(ch%)+"(M*KEY9|M)")
2840 PROCview
2850 ENDPROC
2860 :
2870 DEF PROCview
2880 *FX 138,0,137
2890 *FX 12,6
2900 *FX 11,15

```

VIEW : A Dabhand Guide

```
2910 *WORD
2920 ENDPROC
2930 :
2940 DEF PROCoscli(cline$)
2950 $buf%=cline$
2960 X%=buf%
2970 Y%=buf% DIV 256
2980 CALL $FPF7
2990 ENDPROC
3000 :
3010 REM Error Handling
3020 VDU26
3030 MODE 7
3040 CLOSE#0
3050 REPORT
3060 PRINT" error at Line : ";ERL
3070 PRINT'"Press any key to continue"
3080 KEY=GET
3090 RUN
3100 END
3110 :
3120 DATA "VIEW Manager"
3130 DATA"Load Text","LoadT","This allows you to specify a
text file stored on this disc which will then be
loaded into View ready for you to use. "
3140 DATA"ExCat","ExCat","This option will provide you with
an extended catalogue of your View files listing any
comments placed at the top of the text using the CD
stored command. "
3150 DATA"Template","TEMP","This option will select VIEW
and load in the template of your choice. For example a
dummy letter or report. "
3160 DATA"Precis","PRECIS","Reads in information about
files stored in the PRECIS file. See 'View: A Dabhand
Guide' for full details. "
3170 DATA""
3180 :
3190 DATA "Template Dictionary"
3200 DATA "Letter","T.Plate1","A template fpr a standard
continuous stationary based letter."
3210 DATA "A4 Letter","T.Plate2","A template for a letter
printed on A4 paper"
3220 DATA ""
```

Listing 15.2 - VIEW Manager.

16 : ViewSpell

A fundamental feature of any wordprocessor is that they allow text to be edited and re-edited as many times as you like, whereas typewriters do not: everyone is familiar with the scene of the writer making a mistake, pulling the sheet from the carriage and screwing it into a tight ball before scoring a 'basket'.

Wordprocessors allow you to get your thoughts down quickly, and at that time you don't have to worry too much about grammar or spelling, since these can be corrected once the text is 'on paper'. When typing at speed, letters tend to get transposed, and extra letters are accidentally added whilst others are missed out. And of course we all make spelling mistakes.

In these instances a spelling checker such as ViewSpell becomes invaluable as it will provide you with a list of 'unrecognised' words and allow you to search its dictionary of words for the correct spelling. ViewSpell also has another useful capability - it can be used with ViewIndex to make generating a book or document index a much easier task.

If you have never used a spelling checker, you may be under a slight misconception. A spelling checker can only check spelling: it cannot change and correct incorrectly spelt words for you. ViewSpell contains a dictionary of some 70,000 words on disc, and it checks each word it finds in a VIEW file against this dictionary. If it cannot find a similar word in the dictionary it remembers the word. When the entire document has been checked the unrecognised words are listed. Of course many of these words may and probably will be correctly spelt but simply not in the dictionary - they might be technical terms or proper names - and as such can be ignored (if you know them to be correctly spelt) or checked against a conventional dictionary. Words that are incorrectly spelt can be edited in the normal manner directly in VIEW.

Checking a File

Before you use the ViewSpell disc you are strongly advised to make a working copy of the dictionary disc supplied with the package. This is done by using the *BACKUP command, on which you will find full details in your User Guide (Master and Master Compact) or in your DFS or ADFS manual (BBC B and BBC B+). When the backup copy has been created label it carefully 'ViewSpell Working Disc' and place the original somewhere safe.

Before a document can be spell checked it must of course be written! Once it is finished, save a copy to your normal working disc and then a copy to the ViewSpell working disc. This is now ready to be checked.

With the ViewSpell ROM correctly installed you enter ViewSpell by typing:

```
*SPELL
```

from Command mode. The screen display will change to ViewSpell's Command mode screen, which looks quite like VIEW's and conveys the following information:

```
Bytes free xxxxx  
Mode n
```

where xxxxx is a number. Like VIEW, MODE can be selected and this will alter the bytes free count in all but shadow memory modes - Mode 3 (or 131 on a Master or B+) is normally best and this is selected by:

```
MODE 3
```

The ViewSpell prompt, like VIEW is => but unlike VIEW, ViewSpell does not have an Edit mode screen and pressing ESCAPE has no effect other than printing Escape on the screen.

The file to be checked is now ready to be loaded in. The syntax is the same as for VIEW:

```
LOAD <filename>
```

where <filename> is the name of the file.

ViewSpell loads files in more slowly than VIEW so it prints a series of dots as the file is being loaded - the longer the file the longer a load operation takes and the more dots are printed. The loading sequence is slower so that VIEW has time to process the words in the

file being loaded. When the file has been loaded ViewSpell provides two items of information,:

```
xxx words
xxx unique words
```

The xxx's are of course both numbers. So ViewSpell tells you the number of words in the file and the number of unique words. The former is straightforward. To allow ViewSpell to check more efficiently and quickly it keeps just a list of each time a word occurs in a file. For example words such as 'the', 'to', 'and' and 'then' are very common and will appear many times within a document - for the purpose of checking ViewSpell needs to keep a note of one of each and this is the unique word. You may of course have misspelt 'the' as 'teh' but the latter would be seen by ViewSpell as another unique word and a copy of it would be kept. Figure 16.1 shows a typical ViewSpell screen after the LUNAR text has been loaded in as seen on my setup.

```
ViewSpell
Bytes free 5294
User dictionary 11.0.general
Source :0.Lunar
67 words
Screen mode 3

95 words
67 unique words
=>
```

Figure 16.1. The ViewSpell screen after text is loaded.

A list of the unique words can be seen at this stage by typing:

```
LIST
```

and the list will be displayed on screen. ViewSpell has also sorted the list of unique words into alphabetical order - again to speed up

its checking procedure. If the document is rather long then there is likely to be a long list of unique words. The list will then be much too long to see on screen in one go so it is worth placing ViewSpell in Paged mode - this is done by pressing CTRL-N. Now when the screen is full, scrolling will stop until you press the SHIFT key which causes a new screen of words to be listed. Paged mode can be turned off with CTRL-O.

The Spell Check

To begin checking, the dictionary disc must be in the disc drive. All you need to do is to type:

CHECK

The disc will whirl and the checking process will begin. This can take several minutes for long documents and so a series of dots is printed as the checking progresses. Be patient!

At the end of the checking, ViewSpell will print the number of words that it has not recognised. If 43 words were not found then it would display:

43 words not found
User dictionary?

You can create your own dictionaries of words, and these are called user dictionaries. ViewSpell at this stage prompts you as to whether or not you wish to use a user dictionary to check any extra words. We will examine these later, but for now simply pressing the RETURN key will end the spell checking process.

Typing LIST will show you all the unrecognised words. They can also be printed out by first activating with CTRL-B then typing LIST followed by CTRL-C to turn the printer back off.

The unrecognised words can also be saved to disc using the SAVE command with a given filename:

SAVE <filename>

Spelling Check

The words in the unrecognised list will fall into three categories: a) those that are correctly spelt but not in the dictionary, b) those that are obvious spelling errors and c) those that are incorrectly spelt but you are not sure of the spelling.

The first two categories are easy to solve - for the third we must resort to a dictionary. Of course there is a dictionary of some 70,000 words on the ViewSpell disc and we can search this. But how do we locate the correct spelling of a word if we don't really know how to spell it? Well, just like using a paper dictionary, we get as near to it as possible and then look at all the similar spellings until we locate the correct one. This is where wildcards come into play.

A wildcard is a character which can be any character. Just as in the card game Pontoon, where Douces or Jokers can be wild, there are two types of wildcard, the * and the ?. The command to search through the dictionary is SEARCH and this is followed by the ambiguous search word. If we wished to locate the correct spelling of Exiguity and could only remember the first four letters then we could use the command:

```
SEARCH Exig*
```

and this will display all words in the dictionary beginning with Exig, of which there are 6 in all.

A more concise check can be made using the ? character which can be used to stand wild for single characters. Using the same example, if we knew that Exiguity had 8 letters the first 4 of which were Exig and the last was y then:

```
SEARCH Exig???y
```

would give us a perfect match, which can then be noted.

Using the SEARCH and note technique the correct spellings can be obtained.

If at any time you wish to see the document that is undergoing the spell check, just ensure the disc containing the file is in the drive and then type:

```
SCREEN <Filename>
```

Marking the Document

The next step in the process is to correct the spellings in the original file. One way to do this would be to enter VIEW, load the document and then use the Command mode command SEARCH with the incorrect spelling given by ViewSpell, then enter Edit mode and use CTRL-F1 NEXT MATCH to locate each error. A more efficient way is to use ViewSpell's ability to mark the text file on disc, saving it as a

separate file. This is done with the MARK command for which the syntax is:

```
MARK <filename>
```

You can prefix or end the filename with M to remind you that the file is a marked version. Thus if your filename is "Plan", the marked file might be saved as:

```
MARK PlanM
```

When you hit RETURN, ViewSpell will begin the marking process, which will take a few moments depending on the number of unrecognised words. As with LOAD and CHECK, ViewSpell will display a series of dots after a 'Marking' prompt as the process takes place.

Once the marking has completed the marked file can be loaded into VIEW as follows:

```
*WORD
MODE 3
LOAD PlanM
```

ViewSpell uses a hash and exclamation mark as its markers, ie:

```
#!
```

so to locate each marked word we type:

```
SEARCH #!
```

from Command mode. VIEW will then locate the first marker and you can alter the spelling of the marked word if desired, simply by re-typing it and deleting the misspelt word or by selecting Insert Mode and overtyping the misspelt word. Pressing CTRL-11 will move the cursor onto the next marked word which can be altered and so on until the entire text is corrected. Of course we now have numerous markers within the document which are not needed. To remove them simply type the following in Command mode:

```
REPLACE #!
```

Pressing RETURN after the #! will make VIEW replace each occurrence of the markers with nothing, effectively deleting them. The new, correctly spelt, document can then be saved over the original and is ready for printing.

Even before a document is marked and saved to disc the unrecognised words can be examined in the context in which they are used. Typing:

```
CONTEXT
```

will display the whole line containing each incorrectly spelt word. As the original document file must be read from disc, the disc containing the file must be present in the drive. The CONTEXT command may be followed by an unrecognised word, and in this case only the line or lines containing that word will be displayed.

Disc Management

The above example assumes that the document to be spell checked will be saved onto the ViewSpell working disc. This is not always convenient as it requires files to be copied onto the disc, and if you are planning to create your own user dictionaries disc space may be tight. If you have dual drives then you can use a prefix code to inform ViewSpell where it can locate various components such as the dictionary, user dictionary and text. This is done with the PREFIX command. As an example, consider that we have dual double sided disc drives and we wish to use the ViewSpell dictionary disc in drive 1 and our VIEW text disc in drive 0. The PREFIX command must be followed by one of 3 letters which have the following meaning:

```
T - Text
M - Master
U - User
```

where Text is the document text, Master is the main Master dictionary and U is a User dictionary (more on which later). So the syntax to obtain Text from the disc in drive 0 and Master and User dictionaries from drive 1 would be:

```
PREFIX T :0.
PREFIX M :1.
PREFIX U :1.
```

Now when you tell ViewSpell to load a file, ie:

```
LOAD Plan
```

it will add :0. to the front so that the Load operation will actually perform:

```
LOAD :0.Plan
```

If you are using ADFS then the drive number and a directory root may also be given. If Plan was being stored in a directory called TEXT which itself was held in REPORT the PREFIX command would be:

```
PREFIX T $.REPORT.TEXT.
```

Now when 'Plan' is loaded, ViewSpell will actually perform:

```
LOAD $.REPORT.TEXT.Plan
```

Typing PREFIX at any time will give you a list of the current file prefixes.

!Boot Files

If you get into the habit of using the same working conditions then all these and other ViewSpell settings can be included on your disc as a !Boot file. This can be created in the manner described for VIEW !Boot files. A typical !Boot file for ViewSpell might look like this:

```
*SPELL
*TV0,1
*TV 255,1
MODE 3
PREFIX T :0.
PREFIX M :1.
PREFIX U :1.
```

Of course you just tailor it to your own needs.

Markers

The choice of #! as a marker is made because it is unlikely that you will use these two symbols side by side in a document. But there may well be a time when this does happen, or you may just not wish to use them for whatever reason, so it is possible to change the marker character using the MARKER command. ViewSpell does in fact have two markers, though by default only marker 1 is used to and this is at the front of the unrecognised word. Marker 2 may be used to place the marker at the end of the word, ie, the word can either be enclosed by markers at each end or may have the marker placed either at the start or the end of the word. To set a marker you used the MARKER command as follows:

```
MARKER <n> <x>
```

where n is the marker number and x the marker character(s) to be used by it. So to make ViewSpell mark all text at the front only using the | character, the command would be issued as follows:

```
MARKER 1 |
```

To enable marker 2, it has to be defined, ie,

```
MARKER 2 |
```

To stop any marker being used then don't specify the marker character after the command. We could turn marker 2 off with:

```
MARKER 2
```

Highlight characters can even be used as markers if you wish. Precede the highlight symbol (- or *) with a hat character, ie, ^. This is useful if we wish to print the marked file out. For example, to underline misspelled words markers 1 and 2 must be set as follows:

```
MARKER 1 ^*
```

```
MARKER 2 ^*
```

The current marker settings can be seen at any time from Command mode by simply typing:

```
MARKER
```

If you are limited to just a single disc drive then you may wish to save the marked text file onto another disc - this can be done by using the command CMARK:

```
CMARK PlanM
```

where PlanM is the name under which the marked file will be saved. You may need to swap discs back and forth a few times. When ViewSpell wishes to read more from the text disc it will prompt you to:

```
Insert <filename> disc & hit a key
```

and then prompt you again to replace the destination disc.

If you wish to compile a list of unique words from a whole series of files then you can by using the READ command. When you LOAD a new file, ViewSpell erases all the unique words from its memory and starts afresh. The first file in your series should be LOADED as usual but all subsequent files should be READ in, ie:

```
READ <filename>
```

After which a new word count will be displayed.

If you are loading a very long document into memory then it may be too long for ViewSpell to store all at once, in which case the error message:

```
Memory Full
```

will be displayed. More memory can be made available by changing to a lower screen mode, ie,

```
MODE 7
```

and then reloading the file. If you have a BBC B+, Master Compact or Master 128 then shadow memory can be evoked to gain more room, ie,

```
MODE 131
```

User Dictionaries

The dictionary that is supplied with ViewSpell is the Master or general dictionary and contains about 70,000 of the most common words in the English language. If you are a specialist in a certain area, for example a surgeon or doctor, it is unlikely that medical terms such as Urology, Vasectomy and Epididimovasostomy will be in the Master dictionary, so that each time you spell check a document these words will not be recognised and will be listed as such. This is where a User dictionary comes into play. You can create a dictionary full of the unusual words that you use. As we have seen, when the Master dictionary has been checked ViewSpell prompts to see if you wish to use a User dictionary - if you do then this will be checked for words not so far recognised.

A User dictionary is constructed in two stages. The first stage is to use the CREATE command to write the name of the dictionary to disc. The User dictionary will be created on the disc defined by the PREFIX U command. The filename used in the CREATE command should reflect the contents of the dictionary, so if our doctor is a Urologist the command might be:

```
CREATE Urology
```

If the PREFIX U setting was :1, then the Urology dictionary would be created on the disc in drive 1.

Other User dictionaries can be created in a similar fashion. Computer jargon might be common in your document, so a dictionary called COMPUTER or MICRO might be of use, ie,

CREATE Micro

The name of your dictionary will of course be restricted to the number of characters allowable for a filename by your filing system, ie seven for DFS and ten for ADFS.

Adding Words

CREATE reserves and format an area on the disc surface and assigns to this the filename specified after the CREATE command. Now the dictionary must have words added to it. You can do this in two ways. Either sit down and add them all at one go using a suitable reference book, or else add each word as and when it is signalled as an unrecognised word by ViewSpell. In each case the procedure is the same. First we use the AW (Add Word) command to specify the dictionary to which words are to be added, ie,

AW Urology

ViewSpell will check to ensure that the dictionary exists and then prompts the question:

word?

Now you simply type, carefully, the word to be added and press RETURN. Before adding the word to the dictionary, ViewSpell first checks the Master dictionary to ensure that you are not duplicating the addition. If the word is not found it adds it to the User dictionary, otherwise the message 'In master dictionary' is displayed and the word is not added. In either case the next word? is requested. When you have added enough words simply press ESCAPE. The new words in the User dictionary can then be displayed if you wish by the SEARCH command as follows:

SEARCH * Urology

At some time you may wish to delete a word from the dictionary; perhaps you spelled it incorrectly! The process is exactly the same except the DW (Delete Word) command is used, ie:

DW Urology

word? is then prompted, but in this case it is the word to be deleted. If the word is found and deleted the message 'Deleted' is printed.

The CHECK command can be limited to a User dictionary simply by specifying the dictionary name after the CHECK command, ie,

CHECK Urology

The User Dictionary

After the Master dictionary has been checked, ViewSpell prompts:

```
User dictionary?
```

In the earlier example we said 'No' to this prompt by pressing the RETURN key, however if we had said YES, ViewSpell would then automatically check the assigned User dictionary automatically for unrecognised words. The User dictionary must of course be assigned and this is done with the USER command:

```
USER Urology
```

This could be included in the !Boot file.

ADDing Words

The ADD command is a much quicker way of building up a User dictionary. You need to make up a VIEW file containing all the new words you wish to add to the dictionary. Then LOAD this into ViewSpell in the normal way and CHECK to ensure that only the new words are added to the dictionary. Then simply type:

```
ADD
```

and the list of words not recognised will be added to the User dictionary one by one. As each word is listed you are required to answer with a keypress as follows:

```
Y <RETURN> add the word
N <RETURN> no don't add the word
D <RETURN> delete the word from the list
^ <RETURN> go back one word
<ESCAPE> stop adding words
```

If you wish to add the words to another User dictionary, MICRO for example, then the command syntax becomes:

```
ADD MICRO
```

In both cases the dictionary disc must be present and in the correct drive as defined by the PREFIX command. If you are quite sure that you wish to add all the words in your file then you could make another file similar to a !Boot file, which written in VIEW might look like this:

```
USER Urology
LOAD uwords
```



```
CHECK
ADD
Y
Y
Y
Y
Y
Y
Y
```

The number of Y's should be the same as the number of words to be added to the User dictionary. This should be saved with WRITE using a suitable filename, ie,

```
WRITE makeU
```

and can be 'run' with the *EXEC command:

```
*EXEC makeU
```

Indexes

As described in the next chapter, ViewSpell can be used in conjunction with ViewIndex to compile an index. ViewSpell can also be useful in creating indexes for books or documents that may have been written in VIEW but then typeset by more traditional methods so that page numbers have no real meaning. I recently used the following technique to compile a fully cross referenced index of over 700 entries in less than a day! With final proofs of your document in front of you, enter VIEW and then enter each index item and page number as a single word, ie, no spaces whatsoever. If for example on page 43 of the typeset document a reference was made to the 'BASIC Interpreter', two entries into VIEW could be made as follows:

```
BASICInterpreter43
InterpreterBASIC43
```

Continue in this way until the complete index list has been made, and save it to disc. Then enter ViewSpell and load the file. Even though you may have several entries for the BASIC Interpreter, because page numbers change each one will be seen by ViewSpell as a unique word. As ViewSpell loads the file it also sorts it into alphabetical order for you. Once complete, simply SAVE the list of unique words and then load it back into VIEW to insert the spaces and do any general sprucing up.

17 : ViewIndex

ViewIndex is supplied on disc and is a suite of programs and files that allow you to build up an index of words or phrases. The index may take two forms, either a number index which uses page number, ie,

`wordprocessor 45,67,89`

or a sectional number index, which references chapter, section and sub section,

`wordprocessor 1.2, 1.3.4, 5.1.3`

In the former case the Index program is only of use if you are typesetting your document directly from VIEW. If you are using VIEW to prepare a document that will be set by traditional means then it is unlikely that page numbers in VIEW will tally up with those of the final typeset copy and hence ViewIndex is of little use in such cases the second index is of great value. The previous chapter on ViewSpell outlines a method for preparing VIEW index files of this type by using a combination of VIEW and ViewSpell.

There are four stages which you must go through to create an index. These are:

1. Preparing a marked text file
2. Create a series of intermediate indexes
3. Create the ViewIndex file
4. Editing the ViewIndex file

Let's look at each stage in turn.

Stage 1 - Marking Entries

For ViewIndex to create an index it must be told which words are to be included in the index. This is done by enclosing each word within markers. In ViewIndex the markers are double highlight 1 codes (SHIFT-14), so if the word BASIC is to be included in the index, each occurrence of it must begin and end with two sets of highlight 1:

```
-- BASIC --
```

where **-** is obtained by pressing SHIFT-I4.

Index entries are not limited to single words. It is quite possible to include a phrase such as 'BASIC Interpreter'. To do this just position the first set of double highlight 1 codes before the 'B' in 'BASIC' and the second set after the last 'r' in 'Interpreter' thus:

```
-- BASIC Interpreter --
```

The SEARCH command can, as outlined in Chapter 5, be used to locate all occurrences of a particular word or phrase. But there is a limit to the number of characters that can be included within a single index entry - the magic number is 50.

Manually searching through a document and marking certain words is both time consuming and somewhat tedious - if you have ViewSpell then the whole process of marking can be automated. This is what you do.

First create a list of words that you wish to be included in the index. This is best done by making a list of unique words in the document. If the document is spread across several files then the second, third and other files can be READ in as described in the previous chapter. This file will contain quite a lot of words that you do not want in the index so the list should be saved, loaded into VIEW, edited and then saved back to disc before being reloaded into ViewSpell. Markers 1 and 2 should then be redefined as double highlight characters:

```
MARKER 1  ^-^-
MARKER 2  ^-^-
```

The disc containing a copy of text to be marked should then be inserted into the drive and marked using the MARK command. If there are several files which make up the document then each one will need to be marked in turn - remember to give each marked file a distinctive name. You should end up with a series of copy files with all of the words to be included in the index, marked by double highlight 1 codes as required, in a fraction of the time it would take to do manually. For the purpose of this text we will assume that we have three document files, now marked and stored on disc with filenames: Plan1M, Plan2M and Plan3M.

Highlighting Problems

The highlight 1 command is normally used by VIEW to indicate to a suitable printer driver that the text following should be underlined up until the next highlight 1 character. It is quite likely then that our VIEW files to be indexed will contain highlight 1 characters around words and phrases. This is no great problem because by using a double highlight 1 code the highlighting effect is turned on and then off before it starts. For example the word 'Dabhand' might be underlined in the text as follows:

- Dabhand **-**

Placing double highlight 1 commands around it will give:

--- Dabhand **---**

The first two highlights mark the word, the third turns underlining on. The next highlight will turn the underlining off while the final pair will act as the marker.

Highlight 2 must be treated with a little more respect otherwise it may be interpreted as an extended highlight sequence to switch on an alternative font, rather than bold followed by a ViewIndex marker. The way around this is to ensure that ViewIndex double highlight 1 markers always come before a bold highlight 2 command, ie, **--- *** or alternatively to leave a space between a bold highlight and the ViewIndex highlights.

Stage 2 - Creating Intermediate Index

The next stage is to create a file or series of intermediate files which contain the words and page/section numbers from each marked document file. This is done by reading the marked file(s) through a special template, which can be thought of as a VIEW macro. There are two standard ones supplied on the ViewIndex disc itself. These files are:

TPP - Template for page numbers

TPS - Template for section numbers

The TPP file will be used in the following pages to create an index with page numbers.

Enter VIEW and load a printer driver from the ViewIndex disc called Index. This is done as follows:

***WORD**
PRINTER Index

The Index driver only looks like a printer driver to VIEW; in fact it cheats and instead of directing text to the printer it will direct it to a disc file. We now need only to PRINT the TPP template macro and each of the marked text files as follows:

PRINT TPP Plan1M Plan2M Plan3M

VIEW will print each file in turn, but the Index printer driver will look out for the ViewIndex double highlight code and send the words they embed with the current page number (extracted from register P) to disc. This process may take a few moments depending on the length and number of the document files.

When you have finished, a special intermediate index file, called I.INDEX, will exist on the disc containing words and page numbers in the order in which they were encountered in the document file(s). It may be that you have a few other files which make up the document, and these can be processed in exactly the same way, but before you go on you must rename the I.INDEX file otherwise it will be overwritten. Changing the name is easy - think of a new name, ie, I.PlanA - and use the *RENAME command:

***RENAME I.INDEX I.PlanA**

Your User Guide or DPS/ADPS manual contains full information on the *RENAME command.

Stage 3 - Creating the Main Index

The names in the intermediate index file(s) must now be sorted into alphabetical order. To do this Boot the ViewIndex disc by pressing the SHIFT-BREAK keys together - a menu of 4 items will be presented on the screen:

1. Create Index
2. Merge Indexes
3. Enter VIEW
4. End program

Pressing key 1 to 4 will select the appropriate option - at the moment we want to create the index so option 1 is required. The index creator and sorting program will then be loaded and you will need to be on hand to answer some questions. The first prompt is:

VIEW : A Dabhand Guide

```
Insert disc containing  
intermediate index file  
and enter filename:
```

So you should replace the ViewIndex with the disc containing the recently created intermediate index file(s). If you have renamed the file as above then enter this filename, but if not then you can just press RETURN and the LINDEX file will be looked for automatically. The next prompt is:

```
Enter filename  
of final index:
```

Of course this can be anything you wish, but the convention is to use directory O for final indexes so a suitable filename might be:

```
O.Plant
```

If you press RETURN then a default filename of O.INDEX will be used. The next prompt will be:

```
Ignore case? (Y/N)
```

This acts a bit like VIEW's FOLD command. If you respond with Y(es) then words such as 'VIEW', 'View' and 'view' will be seen as the same word and only a single entry will be created in the index for all three. If you respond with N(o) then 'VIEW', 'View' and 'view' would be treated as three different items and each would have a separate entry in the index.

```
Justify reference numbers? (Y/N)
```

Reponding Y(es) to this prompt will justify line numbers to the right of the index entry. Pressing N(o) will make ViewIndex place line numbers directly after the index entry. The following two examples show a typical entry with line numbers justified right and then without justification respectively:

```
Stored Commands                      34,56,57,67,89,99  
Stored Commands 34,56,57,67,89,99
```

The next prompt allows you to set a line length limit:

```
Line length ?
```

This can be from 10 to 80 characters across - just think of it as a line of text in VIEW. What number you enter really depends on your index. If in doubt always go for the standard ruler width in your document, say about 50 to 60 characters.

The final prompt determines how an index entry will begin:

Should the first letter
of each entry be:

1. upper case
2. lower case
3. unchanged?

This allows you to set a style. Pressing each option in turn will make the word 'View' appear as follows in the index:

1. View (the V is forced to upper case)
2. view (the V is forced to lower case)
3. View (the entry is used as supplied in the text)

You will then be prompted to enter a disc in the drive which is to be used to save the final index on - and then the final index is constructed, which may take a few minutes depending on the number of entries. During the procedure messages are displayed on screen to tell you what's happening:

```

Reading  Index (reading intermediate file from disc)
Sorting  Index (putting entries into alphabetical order)
Merging  multiple entries (ie, if VIEW occurs several times it
                        will make it into one entry)
Writing sorted index (saving the final index to disc)
Finished

```

The number of entries in the index will then be displayed.

Stage 4 - Editing the Index

Finally, you should load the index into VIEW and edit it to suit your own requirements. If the index is very long then the EDIT command can be used to process the file continuously. Use of the CHANGE, SEARCH and REPLACE commands will help in the editing process. Multiple references can be combined even further if so desired: for example with this book there will be numerous stored command entries thus:

```

Stored Command CE 54,56,78
Stored Command LM 56,57,79
Stored Command LS 55,56,98,99

```

These could be edited down to:

Stored Commands:

- CE 54,56,78
- LM 56,57,79
- LS 55,56,98,99

Once the final index is ready it can be saved and then printed.

Indexing by Sections

If you have read and understood Chapter 12 then you should have little difficulty in using the section numbering index. The TPS file on disc contains the macros for incrementing chapter, section and subsection numbers. The macros are called:

```
IC Increment Chapter
IS Increment Section
IT Increment sub-section
```

You simply insert these as stored commands at the start of each chapter, section and sub-section. It is worth loading the TPS file into VIEW and examining it in conjunction with reading Chapter 12.

Master owners will need to alter the TPS file as it uses register IT to hold the sub-section number, which is already used on the Master to print the time. References to this should be changed in macro IT to something suitable, such as B. The T at the top of the TPS file (before I,INDEX) should also be changed accordingly.

To create the index you simply follow all the stages above except that the TPS file should be printed instead of the TPP file:

```
PRINT TPS file1 file2
```

18:VIEW Utilities

Presented in this chapter are a few utility programs for use with VIEW. The programs are varied and all have a good practical use. They are:

- * Layout Driver - provides a 'printed' layout on screen
- * Recover - allows you to recover text after NEW
- * File Checker - checks a file with memory
- * Highlight Checker - shows you if highlights are balanced
- * VIEW Help - *HELP for VIEW

The first four utilities can be used on any BBC or Master machine, but the final help utility needs Sideways RAM to run. All five are on the Program Disc.

Utility 1 - The Screen Layout Driver

Throughout this tome I have stressed the importance of layout and presentation of text. VIEW's SCREEN command will give you a good indication how pages will print out; in particular it will show you where page breaks will occur and whether left and right margins are correct. It does not show any highlights in the text which can spoil the usual screen formatting, particularly with tables or a justified right margin, because they take up space on the screen just as they do in text mode.

The program given in Listing 18.1 offers a way of checking the look of each page before printing by providing what can best be called a 'picture' of it!

The Preview Program

Listing 18.1 contains both BASIC and assembler - just enter it as it is and save it with a suitable filename such as BPAGE. The assembler section of the program has a checksum routine within it to ensure that the machine code it generates is correct. If after running the program you get any errors check the listing, correct and save again. If you have a Master 128, Master Compact or a BBC micro with the

Graphics Extension ROM (GXR) fitted then change line 190 as follows:

```
190  gxr=TRUE
```

If your BBC micro does not have GXR fitted then do not alter the listing. When correct the program will save the machine code it generates as a file called V.SCREEN. If you are using ADFS then you must first create the V directory, ie:

```
*CDIR U
```

or change line 260 to a suitable filename using a directory that already exists. The file created is a VIEW printer driver. In VIEW command mode, it can be loaded with:

```
PRINTER V.SCREEN
```

Once loaded, V.SCREEN will work in any of the graphics modes 0, 1, 2, 4 or 5, or their shadow equivalents on a B+ or Master 128 and Master Compact. If the display mode is either text-only or teletext (3, 6 or 7), then it must be changed to one of the graphics modes (it doesn't matter which one) ie:

```
MODE 4
```

Now any document can be previewed with either of the PRINT or SHEETS commands as normal. For example, to preview a VIEW file called CHAP18, type:

```
SHEETS CHAP18
```

V.SCREEN draws the outline of each sheet of 'paper' on the screen, then adds the words in the form of small black bars rather than individual letters. Obviously the words can't be read, but the overall proportions of the page can be judged. Now you can check the page layout, for example the relative sizes of top and bottom margins, the shapes of the paragraphs, and how tables and figures fit in. Figure 18.1 shows an example page dumped from the screen.

Next page..

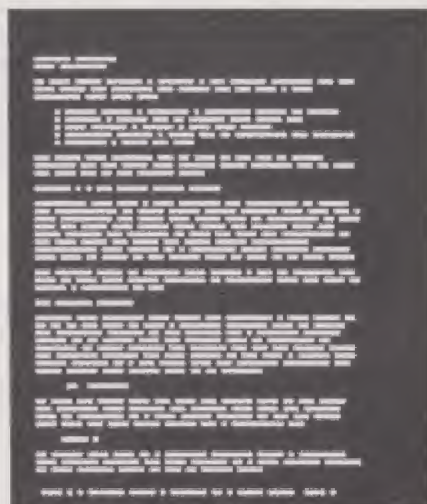


Figure 18.1 - A typical display using V.SCREEN.

If you are previewing with PRINT then you can move onto the next page of displayed text by pressing the space bar. If you have used SHEETS, then press M or space at the 'Page n' prompt. Pressing ESCAPE at any time returns the program safely to VIEW command mode.

A Stationery Difference

V.SCREEN will display text as it would appear when printed in Pica on a sheet of A4 paper. This layout is only suitable if you are using A4. If you are using continuous stationery then it will be necessary to assemble a new version of V.SCREEN. Of course you can assemble several different versions as you wish - though do remember to use new filenames for each. To assemble a new version, changes can be made to the values given to the constants 'lines' and 'chars' in Listing 18.1, line 200. These constants tell the program the number of lines per page and the number of characters across the whole width of the paper (not just the length of the ruler).

Let's take the example of normal listing paper. This is 9.5 x 11 inches, with 0.5 inch tear-off strips either side for the sprocket holes. This gives space for 82 Pica characters across and 66 lines down. The dimensions of other common stationery are given in the table below. The screen display can be up to 82 lines of 160 characters: if the paper is too big, the program still works but only the central portion of the sheet is shown.

To assemble this new version, the new dimensions should be inserted into line 200, and the file name changed in line 260. The new lines then will read:

```
200 lines=66 : chars=82
260 name$="V.ListScr"
```

Common Paper Sizes - assuming 1/6th inch line spacing and 1/10th (Pica) or 1/12th inch (Elite) character spacing.

Lines	Width in Chars	Pica	Elite
9.5 x 11	66	82	102
14.5 x 11	66	135	162
A5 (portrait)	50	59	70
A5 (landscape)	35	82	99
A4 (portrait)	70	82	99
A4 (landscape)	49.5	117	140
A3 (portrait)	99	117	140
A3 (landscape)	70	165	198

Early VIEW

Early versions of VIEW (ie 2.1 and 1.4) will clear the screen on the completion of every command, which will make it difficult to preview the last sheet of a document. If you find this happening then simply insert two extra page eject stored commands at the end of your documents.

Utility 2 - The VIEW Recovery Program

I can say with all certainty that there will come a time when you accidentally delete from View the text you are currently working on by using the NEW command before you have saved your text. Thus it is theoretically lost for all time. The following utility will be of great comfort, as it allows you to locate your VIEW text in memory so that you can re-save it ready for READING back into VIEW. Of course if you

should type NEW and then load in some more text then your lost text has gone forever.

When you type text into VIEW it stores each character in memory. If you should accidentally type NEW it does not erase this but merely resets VIEW's pointers so that it thinks it isn't there. Of course if you should happen to load in new text it is almost certain that the new text will be placed over the old 'lost' text. If this is the case you can only hope that the new text is considerably shorter than the lost text so that you can at least restore some of it, which is better than retyping the whole lot.

Listing 18.2 provides the DUMP program, which is written mainly in assembler - type the program in as you see it. If you make any errors these will be reported. If after RUNning the program gives the message:

```
Checksum error
```

recheck the listing carefully. Once the program has RUN correctly it will automatically save the machine code it generates onto your VIEW Utilities disc. The file name used is VDUMP. You should now save the program listing itself using the filename DUMP.

Dry Run

It is well worth a practice run with the program at this stage, so that you have an idea what to do when you really need to use it. (Note that it's no good trying to type the program for the first time after you've lost the text. It has to be used straight from tape or disc: after all, you can't wait until you crash your car before shelling out for insurance.) First enter some text into VIEW and from Command mode type NEW. For demonstration I am using the LUNAR text.

The first thing when you 'lose' some text is NOT to panic! Find the disc or tape with the VDUMP routine on it. Next enter BASIC by typing

```
*BASIC
```

Once the familiar > prompt appears run the machine code of VDUMP. If you are using disc this can be performed simply by typing the command

```
*VDUMP
```

while tape users will need to use

*RUN VDUMP

Once the machine code has loaded, the screen will clear to show a row of numbers, as illustrated in Figure 18.2. The left hand column is the memory address currently being displayed. This will be the address of PAGE, ie the point where the micro places programs, which on my micro is &1F00 - yours may be different. After the line address there follow eight memory bytes, their contents expressed in hexadecimal notation. After this are a further eight characters. If the memory contents are an ASCII character then these bytes will show the ASCII characters, otherwise a full-stop will be shown.

1F00	AA	00	00	00	00	00	00	00	00
1F08	00	00	00	00	00	00	00	00	00
1F10	00	00	00	00	00	00	00	00	00
1F18	00	00	00	00	00	00	00	00	00
1F20	00	00	00	00	00	00	00	00	00
1F28	00	00	00	00	00	00	00	00	00
1F30	00	00	00	00	00	00	00	00	00
1F38	00	00	00	00	00	00	00	00	00
1F40	00	00	00	00	00	00	00	00	00
1F48	00	00	00	00	00	00	00	00	00
1F50	00	00	00	00	00	00	00	00	00
1F58	00	00	00	00	00	00	00	00	00
1F60	00	00	00	00	00	00	00	00	00
1F68	00	00	00	00	00	00	00	00	00
1F70	00	00	00	00	00	00	00	00	00
1F78	00	00	00	00	00	00	00	00	00
1F80	00	00	00	00	00	00	00	00	00
1F88	00	00	00	00	00	00	00	00	00
1F90	00	00	00	00	00	00	00	00	00
1F98	00	00	00	00	00	00	00	00	00
1FA0	00	00	00	00	00	00	00	00	00
1FA8	00	00	00	00	00	00	00	00	00
1FB0	00	00	00	00	00	00	00	00	00
1FB8	00	00	00	00	00	00	00	00	00

Figure 18.2 - Memory dump from &1F00.

Currently the screen display shows 24 lines of memory, and pressing the space bar once will display the next 24 lines. The idea is to keep displaying memory until you come across your text. VIEW normally starts to store its text at PAGE+&100. You can find this address when in BASIC by typing:

```
PRINT ~(PAGE+&100)
```

On a Master 128 and Master Compact PAGE is normally set to &E00, so the result of the small sum should be &F00 and on my micro it is &2000, ie &1F00+&100=&2000.

Look for this address or the start of a VIEW file which is signified by the ruler - this is shown in figure 18.3. We need to note the address of this somewhere safe. Looking at figure 18.3 we can see that the ruler begins at &2003.

[illegible]

Figure 18.3 - The start of the text to be identified.

The next step is to find the end of the text, and again this just requires you to step through memory until you find it, as shown in figure 18.4. Again calculate the address of the last character you wish to save. Here it's the RETURN character signified by the &0D byte whose address is &225E.

21E0	74	6F	6F	20	66	61	72	20	too far
21E8	74	6F	20	70	72	65	76	65	to preve
21F0	6E	20	61	70	65	74	6B	54	t a tak
21F8	6E	20	66	66	66	2E	61	20	nt-off.
2200	66	20	66	66	66	72	73	66	he first
2208	66	20	66	66	66	72	66	72	word fr
2210	66	20	66	66	66	6E	20	6F	om, man
2218	66	20	66	66	66	20	60	6F	n the mo
2220	66	20	66	66	66	61	60	65	on came
2228	66	20	66	66	66	20	41	6C	from fld
2230	66	20	66	66	66	22	54	72	rin: "Tr
2238	61	6E	71	75	69	6C	69	74	anquil
2240	79	6E	62	61	73	65	2E	20	y base.
2248	54	6E	65	8D	45	61	67	6C	The Eagl
2250	65	20	68	61	73	20	6C	61	e has la
2258	66	64	65	64	22	2E	80	80	nded"...
2260	80	80	80	80	80	80	80	80
2268	80	80	80	80	80	80	80	80
2270	80	80	80	80	80	80	80	80
2278	80	80	80	80	80	80	80	80
2280	80	80	80	80	80	80	80	80
2288	80	80	80	80	80	80	80	80
2290	80	80	80	80	80	80	80	80
2298	80	80	80	80	80	80	80	80

Figure 18.4 - The end of the the text.

Now that we know the start and end address of the text to be restored we can leave the VDUMP program. To do this press ESCAPE. The penultimate step of the restoration process is to save the text. Do this with

```
*SAVE TEXT 2003 225E
```

of course you substitute your own start and end addresses. The text has now been 'restored' and can be safely read into VIEW using the READ command as follows:

```
*WORD
MODE 3
NEW
READ TEXT
```

Now it can be re-saved as normal for subsequent loading.

Utility 3 - File Checker

Text is valuable, particularly if you have spent several hours or perhaps even days putting together some outstanding prose. It is a simple 'must' always to keep a backup of your text on another disc elsewhere. However, whenever you are saving text it is worth

double checking that what is saved on the disc is truly the current document. This program will compare a disc file with the current document held in VIEW.

Listing 18.3 is straightforward and includes an error check on the machine code section of the program. If you get any errors reported then re-check, correct and save. Once RUN and correct the machine code will be saved when you press a key. The machine code will be called FCHECK.

When you have entered your text save it to disc then use the FCHECK program as follows:

```
*FCHECK <filename>
```

where <filename> is the name of the saved text. If the text is called CHAP18 then the command is:

```
*FCHECK CHAP18
```

Assuming the FCHECK program is on the text disc, it will load. Checking may take several seconds for long documents. If the disc file matches the current document then the message:

```
File matches memory
```

will be displayed. If not then you get the error message:

```
File different from memory
```

in which case you should resave the file. If the file is still not correct then save the file to a different disc as it may be that the current disc has a flaw.

Utility 4 - The Highlight Matcher

A frequent source of annoyance is when you print out a document only to find that you have failed to de-select a highlight. This of course means that all subsequent text may be printed in bold or underlined. Listing 18.4 provides the basis of a highlight matcher. It goes through a file and looks for a highlight 1 or highlight 2 code. When it encounters one it prints all the text that follows until the next highlight of the same type, which will be the cancelling code. In this way you can see straight away if you have forgotten any highlight codes.

Enter and RUN the program as described above - again a checksum is included to ensure that at least the machine code part of the program is correct. The machine code file is called HILITE and is used in one of

two ways. The first is to look at the document in memory and is called from Edit mode using:

***HILITE**

The second way is to specify a filename, in which case this is sought from the current disc and the highlighted areas displayed. The syntax is:

***HILITE <filename>**

If the file cannot be found on the disc then the 'Not found' error message is printed.

Utility 5 - The VIEW Help ROM

The last utility is of use to those of you with sideways RAM installed into your machines. This is standard on Master 128's, Master Compacts and B+128's. BBC B owners can use the software provided they have a sideways ROM or RAM board installed.

Put simply the program when RUN will assemble a special program which when loaded into sideways RAM acts like a ROM. It will provide you with extended *HELP messages - initially these are :

***HELP VIEW**
***HELP STORED**
***HELP DIRECT**
***HELP OTHER**

Typing: *HELP VIEW at any time will provide you with a list of *HELP options, which when used will display more information, all of which you can be add to and expand. For example entering *HELP STORED could list details of all stored commands.

Forming the Image

The program, given in Listing 18.5, is mostly assembler and once entered should be saved. Because of its extendable nature a checksum cannot be included so do double check the listing. Once RUN the machine code, or ROM image as it is better called, will be saved to disc as ViewHLP. Once on disc it can be reloaded into Sideways RAM as follows:

Master 128 : *SALOAD ViewHLP 8000 6
Compact : *SALOAD ViewHLP 8000 6 1
BBC B+128 : *SALOAD ViewHLP 8000 W
BBC B : *Refer to your ROM/RAM board manual

In all cases once the file has loaded press CTRL-BREAK and then type:

***HELP**

A list of ROM help messages will appear and among them you should see:

```
VIEW Help ROM 1.00
VIEW
```

As already mentioned, typing ***HELP VIEW** lists current ***HELP** commands that can be entered.

Adding your own

Extending the VIEW Help ROM is quite a simple process - you will of course need the original Listing 18.5. As you can see from the listing (and if you have already tried it out), the list of information provided under **STORED**, **DIRECT** and **OTHER** is minimal. This can be extended by adding more text after the commands' labelled counterparts, namely 'stored' (line 1250), 'direct' (line 1360) and 'other' (line 1450). The syntax is the same in all cases:

```
1341 EQU$ "Text placed inside quotes"
1342 EQU$ 13
```

The line numbers above would ensure that this text is placed under the 'stored' label and as such would be printed out when you enter ***HELP STORED**. You will need to renumber the listing at times (**RENUMBER** command in **BASIC**) so then line numbers will change but the position of text entry remains the same.

You can also add your own new ***HELP** commands. The operation involves four processes:

- * Enter it into ***HELP** table
- * Enter label in text
- * Enter text as above but with **EQU\$ 0** terminating text table
- * Add name of command to **VIEW** table

Let's look at an example. If we wish to add help on highlights which should be printed on receipt of the command:

***HELP HILITES**

First add it to the command table:

```
1221 OPT FHTable("HILITES")
```

next add the labelled entry plus some text terminated by a 0 byte:

```

1531 .hilit es
1532 EQU B 13
1533 EQU S "Highlight Help"
1534 EQU B 13
1535 EQU S "    <HT1> gives underlined text"
1536 EQU B 13
1537 EQU S "    <HT2> gives bold text"
1538 EQU B 13
1539 EQU B 0

```

Finally add the HILITE name to be printed on *HELP VIEW;

```

1631 EQU S "    HILITE"
1632 EQU B 13

```

The new version is now ready and the program can be saved and then RUN and loaded into sideways RAM as described earlier. The Programs Disc contains an extensive VIEW Help ROM ready and waiting to be used, plus *HILITE, *PCHECK, and *VDUMP.

Listings

Listing 18.1.

```

10 REM VIEW PAGE Driver
20 REM by Graham Bell
30 REM (C) Bruce Smith/G.Bell 1987
40 REM VIEW: A Dabhand Guide
50 :
60 PROCsetup
70 PROCassemble
80 PROCchecksum
100 PROCoscli("SAVE " * name$ + " " + STR$(store) + " +100 40C
    400")
110 END
130 :
140 DEF PROCsetup
150 DIM store &FF
160 FOR loop%=0 TO &FF
170 loop%?store=0
180 NEXT
190 gxr=FALSE
200 lines=70:chars=82
210 osbyte = &FFF4
220 oswrch = &FFEE
230 osrdch = &FFED
240 origin = &400
250 offset = store - origin

```

```

260 name$="V.Screen"
270 ENDPROC
280 :
290 DEF PROCassemble
300 FOR pass = 0 TO 3 STEP 3
310 P% = store
320 [
330 OPT pass
340 JMP output-offset
350 JMP newpage-offset
360 RTS
370 BRK
380 BRK
390 RTS
400 .char BRK
410 .line BRK
420 RTS
430 :
440 .error
450 BRK
460 OPT FNequb(128)
470 OPT FNequs("Not graphics mode")
480 BRK
490 :
500 .waitforpage
510 LDX #4
520 JSR text-offset
530 JSR osrdch
540 .newpage
550 LDA #135
560 JSR osbyte
570 CPY #3
580 BEQ error
590 CPY #6
600 BCS error
610 LDX #0
620 STX line-offset
630 .newline
640 JSR text-offset
650 STX count-offset
660 STX count+1-offset
670 RTS
680 :
690 .text
700 LDY data-offset,X
710 LDA data+1-offset,X
720 SEC
730 SBC data-offset,X
740 TAX
750 .loop
760 LDA data-offset,Y

```


VIEW : A Dabhand Guide

```
770 JSR oswrch
780 INY
790 DEX
800 BNE loop
810 RTS
820 :
830 .output
840 STA char-offset
850 TXA
860 PHA
870 TYA
880 PHA
890 LDA line-offset
900 CMP #lines
910 BCC notnewpage
920 JSR waitforpage-offset
930 .notnewpage
940 LDA char-offset
950 BMI return
960 LDX #1
970 CMP #13
980 BNE noteoln
990 JSR newline-offset
1000 INC line-offset
1010 JMP return-offset
1020 .noteoln
1030 INX
1040 CMP #32
1050 BNE print
1060 INX
1070 .print
1080 JSR text-offset
1090 SEC
1100 LDA count-offset
1110 SBC #8
1120 STA count-offset
1130 BCS return
1140 DEC count+1-offset
1150 .return
1160 PLA
1170 TAY
1180 PLA
1190 TAX
1200 LDA char-offset
1210 RTS
1220 :
1230 .data
1240 OPT FNequb(string0-data)
1250 OPT FNequb(string1-data)
1260 OPT FNequb(string2-data)
1270 OPT FNequb(string3-data)
```

```

1280 OPT FNequb(string4-data)
1290 OPT FNequb(stringend-data)
1300 .string0
1310 OPT FNvdu(26)
1320 OPT FNvdu(12)
1330 OPT FNorig(640-chars*4,496-lines*6)
1340 OPT FNplot(4,0,0)
1350 OPT FNpage(gxr)
1360 OPT FNplot(4,0,lines*12-5)
1370 .string1
1380 OPT FNplot(0,0,-12)
1390 .string2
1400 OPT FNdot(gxr)
1410 .string3
1420 OPT FNplot(0,0,0)
1430 .string4
1440 OPT FNequs("Next page..")
1450 .stringend
1460 BRK
1470 ]
1480 count = string1 + 2
1490 NEXT pass
1500 ENDPROC
1510 :
1520 DEF PROCchecksum
1530 sum = 0
1540 FOR I% = 0 TO &FF
1550 sum = sum + I%?store
1560 NEXT
1570 IF (sum <> FNTotal(gxr)) AND (lines = 70) AND (chars =
    82) THEN PRINT "checksum error - please check listing":
    END
1580 ENDPROC
1590 :
1600 DEF FNdot(graphic)
1610 IF graphic THEN GOTO 1690
1620 [
1630 OPT pass
1640 OPT FNplot(2,7,0)
1650 OPT FNplot(0,-7,-7)
1660 OPT FNplot(2,7,0)
1670 OPT FNplot(0,1,7)
1680 ] = pass
1690 [
1700 OPT pass
1710 OPT FNplot(98,7,-7)
1720 OPT FNplot(0,1,7)
1730 ] = pass
1740 :
1750 DEF FNpage(graphic)
1760 IF graphic THEN GOTO 1840

```

VIEW : A Dabhand Guide

```

1770 [
1780 OPT pass
1790 OPT FNplot(4,chars*8-1,0)
1800 OPT FNplot(86,0,lines*12-1)
1810 OPT FNplot(6,chars*8-1,0)
1820 OPT FNplot(86,chars*8-1,lines*12-1)
1830 ] = pass
1840 [
1850 OPT pass
1860 OPT FNplot(102,chars*8-1,lines*12-1)
1870 ] = pass
1880 :
1890 DEF Fntotal(graphic)
1900 IF graphic THEN = 44974 ELSE = 44226
1910 :
1920 DEF FNvdu(code)
1930 = FNequib(code)
1940 :
1950 DEF FNorig(x, y)
1960 pass = FNequib(29)
1970 pass = FNequw(x AND 4FFFF)
1980 = FNequw(y AND 4FFFF)
1990 :
2000 DEF FNplot(code, x, y)
2010 pass = FNequib(25)
2020 pass = FNequib(code)
2030 pass = FNequw(x AND 4FFFF)
2040 = FNequw(y AND 4FFFF)
2050 :
2060 DEF FNequib(byte)
2070 ?P% = byte
2080 P% = P% + 1
2090 = pass
2100 :
2110 DEF FNequw(word)
2120 ?P% = word MOD 256
2130 P%?1 = word DIV 256
2140 P% = P% + 2
2150 = pass
2160 :
2170 DEF FNequs(string$)
2180 $P% = string$
2190 P% = P% + LEN string$
2200 = pass
2210 :
2220 DEF PROCoscli(string$)
2230 LOCAL X%, Y%
2240 DIM X% 4FF
2250 Y% = X% DIV 256
2260 $X% = string$
2270 CALL 4FFF7

```



```
2280 ENDPROC
```

Listing 18.1. The VIEW Page Driver.

Listing 18.2.

```
10 REM VIEW Restore
20 REM (C) Bruce Smith 1987
30 REM VIEW: A Dabband Guide
40 :
50 PROCassemble
60 PROCchecksum
70 *SAVE VDUMP C00 C99
80 END 90 :
100 DEF PROCassemble
110 address=470:lines=473
120 oswrch=4FFEE
130 FOR pass=0 TO 3 STEP 3
140 P%-C00
150 |OPT pass
160 LDA #22:JSR 4FFEE
170 LDA #7:JSR 4FFEE
180 LDA #24:STA lines
190 LDA 418:STA address+1
200 LDA #0:STA address
210 :
220 .memorydump
230 JSR addressprint
240 :
250 .hexbytes
260 LDX #7:LDY #0
270 .hexloop
280 LDA(address),Y
290 JSR hexprint:JSR space
300 INY:DEX
310 BPL hexloop
320 JSR space
330 :
340 .asciibytes
350 LDX #7:LDY #0
360 .asciiloop
370 LDA(address),Y
380 CMP #120:BMI fullstop
390 CMP #180:BCC leapfrog
400 :
410 .fullstop
420 LDA #ASC".
430 .leapfrog
440 JSR oswrch
```

VIEW : A Dabband Guide

```

450 INY:DEX
460 BPL ascilloop
470 LDA #13:JSR &FFB3
480 CLC
490 LDA address:ADC #8
500 STA address:BCC nocarry
510 INC address+1
520 .nocarry
530 DEC lines
540 BNE memorydump
550 JSR &FFE0
560 CMP#27:BEQ escape
570 LDA #23:STA lines
580 JMP memorydump
590 :
600 .escape
610 LDA #47E
620 JSR &FFB4
630 RTS
640 :
650 .space
660 LDA #32:JMP oswrch
670 .addressprint
680 LDX #address
690 LDA 1,X:JSR hexprint
700 LDA 0,X:JSR hexprint
710 JSR space:JSR space
720 RTS
730 :
740 .hexprint
750 PHA
760 LSR A : LSR A
770 LSR A : LSR A
780 JSR first
790 PLA
800 .first
810 AND #40F:CMP #10
820 BCC over:ADC #6
830 .over
840 ADC #48:JMP oswrch
850 ] NEXT
860 ENDPROC
870 :
880 DEF PROCchecksum
890 A%=0
900 FOR N%=&C00 TO &C98
910 A%=A%+?N%
920 NEXT
930 IF A%=16464 ENDPROC
940 CLS
950 PRINT "Checksum error"

```

```
960 PRINT "Please check listing"
```

Listing 18.2. VIEW Restore.

Listing 18.3

```
10 REM VIEW File Checker
20 REM by Dave Atherton
30 REM (C) Bruce Smith 1987
40 REM VIEW: A Dabhand Guide
50 :
60 MODE 7
70 PRINT "Assembling VIEW File Checker"
80 PROCsetup
90 PROCassemble
100 PROCchecksum
110 A$="*SAVE FCHECK "+STR$(M$)+"*"+STR$(end-M$)
120 PRINT A$;"Press a key to save"
130 IF GET THEN OSCLI A$
140 END
150 :
160 DEF PROCsetup
170 M$=4900:zp=480
180 memory=zp+3
190 minus=zp+4
200 handle=zp+5
210 pzp=zp+6
220 len=4D
230 oswrch=4FFEE
240 osascii=4FFEE3
250 osnewl=4FFEE7
260 osbyte=4FFF4
270 osargs=4FFDA
280 osfind=4FFCE
290 osbget=4FFD7
300 ENDPROC
310 :
320 DEF PROCassemble
330 FOR pass=0 TO 2 STEP 2
340 P$=M$
350 [OPT pass
360 LDA #1
370 LDY #0
380 LDX #zp
390 JSR osargs
400 LDY #0
410 .allop
420 LDA (zp),Y
430 INY
```


VIEW : A Dabhand Guide

```
440 CMP #32
450 BEQ aloop
460 CMP #i1
470 BNE filename
480 :
490 EQUB 0
500 EQUB 255
510 EQU$ "Syntax : FCHECK <fsp>"
520 EQUB 0
530 :
540 .filename
550 LDX zp
560 LDY zp+1
570 LDA #i40
580 JSR osfind
590 STA handle
600 LDA handle
610 BNE cont
620 :
630 EQUB 0
640 EQUB 214
650 EQU$ "Not found"
660 EQUB 0
670 :
680 .cont
690 LDA #131
700 JSR osbyte
710 INY
720 STY zp+1
730 LDA #1
740 STA zp
750 :
760 .loop
770 JSR getbyte
780 BCS endfil
790 STA memory
800 LDY handle
810 JSR osbget
820 BCS endfil
830 CMP memory
840 BNE diff
850 LDX #0
860 LDY #0
870 LDA #s81
880 JSR osbyte
890 CPY #i1B
900 BNE loop
910 .finite
920 RTS
930 :
940 .diff
```

```
950 EQU 0
960 EQU 255
970 EQU "File different from memory"
980 EQU 0
990 :
1000 .endfil
1010 JSR print
1020 EQU CHR$13+"File matches memory"+CHR$13
1030 EQU 0
1040 RTS
1050 :
1060 .dinc
1070 INC zp
1080 BNE dinc2
1090 INC zp+1
1100 .dinc2
1110 RTS
1120 :
1130 .getbyte
1140 LDY #0
1150 LDA (zp),Y
1160 PHA
1170 JSR dinc
1180 LDA zp+1
1190 CMP len+1
1200 BNE clear
1210 LDA zp
1220 CMP len
1230 BNE clear
1240 LDA #0
1250 LDY handle
1260 JSR osfind
1270 PLA
1280 SEC
1290 RTS
1300 .clear
1310 PLA
1320 CLC
1330 RTS
1340 :
1350 .print
1360 PLA
1370 STA pzp
1380 PLA
1390 STA pzp+1
1400 .ploop
1410 JSR pdinc
1420 LDY #0
1430 LDA (pzp),Y
1440 BEQ pout
1450 JSR osasci
```

VIEW : A Dabhand Guide

```
1460 JMP ploop
1470 .pout
1480 LDA pzp+1
1490 PHA
1500 LDA pzp
1510 PHA
1520 RTS
1530 :
1540 .pdinc
1550 INC pzp
1560 BNE pdinc2
1570 INC pzp+1
1580 .pdinc2
1590 RTS
1600 :
1610 .end
1620 j:NEXT
1630 ENDPROC
1640 :
1650 DEF PROCchecksum
1660 PRINT""Checking code...."
1670 byte%=0
1680 FOR loop=4900 TO 49F7
1690 byte%=byte%+7*loop
1700 NEXT
1710 IF byte%=28093 THEN ENDPROC
1720 PRINT "Checksum error!"
1730 PRINT"Please correct listing"
1740 VDU 7
```

Listing 18.3. The VIEW File Checker

Listing 18.4.

```
10 REM VIEW Highlight Matcher
20 REM by Dave Atherton
30 REM (C) Bruce Smith 1987
40 REM VIEW: A Dabhand Guide
50 :
60 REM The program CANNOT deal with
70 REM extended highlights
80 :
90 MODE 7
100 PRINT"Assembling Highlight Matcher..."
110 PROCsetup
120 PROCassemble
130 PROCchecksum
140 A$="*SAVE HILITES "+STR%-M%+" "+STR%-(end-M%)
150 PRINTA$
160 PRINT"Press any key to save"
```



```
950 EQUB 0
960 EQUB 255
970 EQU$ "File different from memory"
980 EQUB 0
990 :
1000 .endfil
1010 JSR print
1020 EQU$ CHR$13+"File matches memory"+CHR$13
1030 EQUB 0
1040 RTS
1050 :
1060 .dinc
1070 INC zp
1080 BNE dinc2
1090 INC zp+1
1100 .dinc2
1110 RTS
1120 :
1130 .getbyte
1140 LDY #0
1150 LDA (zp),Y
1160 PHA
1170 JSR dinc
1180 LDA zp+1
1190 CMP len+1
1200 BNE clear
1210 LDA zp
1220 CMP len
1230 BNE clear
1240 LDA #0
1250 LDY handle
1260 JSR osfind
1270 PLA
1280 SEC
1290 RTS
1300 .clear
1310 PLA
1320 CLC
1330 RTS
1340 :
1350 .print
1360 PLA
1370 STA pzp
1380 PLA
1390 STA pzp+1
1400 .ploop
1410 JSR pdinc
1420 LDY #0
1430 LDA (pzp),Y
1440 BEQ pout
1450 JSR osasci
```

VIEW : A Dabhand Guide

```
1460 JMP ploop
1470 .pout
1480 LDA pzp+1
1490 PHA
1500 LDA pzp
1510 PHA
1520 RTS
1530 :
1540 .pdinc
1550 INC pzp
1560 BNE pdinc2
1570 INC pzp+1
1580 .pdinc2
1590 RTS
1600 :
1610 .end
1620 ]:NEXT
1630 ENDPROC
1640 :
1650 DEF PROCchecksum
1660 PRINT""Checking code...."
1670 byte%=0
1680 FOR loop=4900 TO 49F7
1690 byte%=byte%+?loop
1700 NEXT
1710 IF byte%=28093 THEN ENDPROC
1720 PRINT "Checksum error!"
1730 PRINT"Please correct listing"
1740 VDU 7
```

Listing 18.3. The VIEW File Checker

Listing 18.4.

```
10 REM VIEW Highlight Matcher
20 REM by Dave Atherton
30 REM (C) Bruce Smith 1987
40 REM VIEW: A Dabhand Guide
50 :
60 REM The program CANNOT deal with
70 REM extended highlights
80 :
90 MODE 7
100 PRINT"Assembling Highlight Matcher..."
110 PROCsetup
120 PROCassemble
130 PROCchecksum
140 A$=""SAVE HILITES "+STR%-M%+"+"+STR%-(end-M%)
150 PRINTA$
160 PRINT"Press any key to save"
```

```

170 IF GET THEN OSCLI AS
180 END
190 :
200 DEF PROCsetup
210 M%=&900
220 zp=&80
230 star=zp+3
240 minus=zp+4
250 handle=zp+5
260 len=&D:
270 oswrch=&FFEE
280 osasci=&FFE3
290 osnewl=&FFE7
300 osbyte=&FFF4
310 osargs=&FFDA
320 osfind=&FFCE
330 osbget=&FFD7
340 ENDPROC
350 :
360 DEF PROCassemble
370 FOR pass=0 TO 2 STEP 2
380 P%=M%
390 [OPT pass
400 LDA #1
410 LDY #0
420 LDX #zp
430 JSR osargs
440 LDY #0
450 STY star
460 STY minus
470 .aloop
480 LDA (zp),Y
490 INY
500 CMP #32
510 BEQ alloop
520 CMP #13
530 BEQ memory
540 LDX zp
550 LDY zp+1
560 LDA #440
570 JSR osfind
580 STA handle
590 LDA handle
600 BNE loop
610 :
620 EQUB 0
630 EQUB 214
640 EQU$ "Not found"
650 EQUB 0
660 :
670 .memory

```


VIEW : A Dabband Guide

```

680 LDA #131
690 JSR osbyte
700 INY
710 STY zp+1
720 LDA #1
730 STA zp
740 :
750 .loop
760 JSR getbyte
770 BCS finito
780 CMP #41D
790 BEQ hstar
800 CMP #41C
810 BEQ hline
820 .cont
830 TAX
840 LDA minus
850 ORA star
860 AND #1
870 BEQ notout
880 TXA
890 JSR wrch
900 .notout
910 LDX #0
920 LDY #0
930 LDA #481
940 JSR osbyte
950 CPY #41B
960 BNE loop
970 .finito
980 JMP osnewl
990 :
1000 .hstar
1010 PHA
1020 INC star
1030 LDY #ASC "*"
1040 JSR conchar
1050 PLA
1060 JMP notout
1070 :
1080 .hline
1090 PHA
1100 INC minus
1110 LDY #ASC "-."
1120 JSR conchar
1130 PLA
1140 JMP notout
1150 :
1160 .conchar
1170 LDA #17
1180 JSR oswrch

```

```
1190 LDA #0
1200 JSR oswrch
1210 LDA #17
1220 JSR oswrch
1230 LDA #135
1240 JSR oswrch
1250 TYA
1260 JSR oswrch
1270 LDA #20
1280 JSR oswrch
1290 TYA
1300 AND #1
1310 TAY
1320 LDA star,Y
1330 AND #1
1340 BNE notcr
1350 JSR osnewl
1360 .notcr
1370 RTS
1380 :
1390 .wrch
1400 CMP #13
1410 BEQ wrch2
1420 CMP #32
1430 BCS wrch2
1440 PHA
1450 LDA #ASC"|~
1460 JSR oswrch
1470 PLA
1480 CLC
1490 ADC #40
1500 .wrch2
1510 JMP osasci
1520 :
1530 .dinc
1540 INC zp
1550 BNE dinc2
1560 INC zp+1
1570 .dinc2
1580 RTS
1590 :
1600 .getbyte
1610 LDA handle
1620 BEQ nofile
1630 LDY handle
1640 JMP osbget
1650
1660 .nofile
1670 LDY #0
1680 LDA (zp),Y
1690 PHA
```

VIEW : A Dabband Guide

```
1700 JSR dinc
1710 LDA zp+1
1720 CMP len+1
1730 BNE clear
1740 LDA zp
1750 CMP len
1760 BNE clear
1770 LDA #0
1780 LDY handle
1790 JSR osfind
1800 PLA
1810 SEC
1820 RTS
1830 .clear
1840 PLA
1850 CLC
1860 RTS
1870 :
1880 .end
1890 J:NEXT
1900 ENDPROC
1910 :
1920 DEF PROCchecksum
1930 PRINT"Checking machine code..."
1940 byte%=0
1950 FOR loop=1900 TO 19F5
1960 byte%=byte%+?loop
1970 NEXT
1980 IF byte%=30408 THEN ENDPROC
1990 PRINT"Checksum error!"
2000 PRINT"Please check program"
2010 VDU 7
```

Listing 18.4. VIEW Highlight Matcher.

Listing 18.5.

```
10 REM VIEW Help ROM
20 REM requires Sideways RAM
30 REM (C) Bruce Smith 1987
40 REM VIEW: A Dabband Guide
50 :
60 PROCsetup
70 PROCassemble
80 *SRWRITE 5000+200 0000 5 I
90 END
100 :
110 DEF PROCsetup
```



```

120 osnewl=4FFE7
130 osasci=4FFE3
140 comline=4F2:vector=4A8
150 Ystore=4AA
160 title$="Help "
170 DIM char% 20
180 ENDPROC
190 :
200 DEF PROCAssemble
210 FOR pass=4 TO 7 STEP 3
220 P%=48000:O%=45000
230 |opt pass
240 BRK:BRK:BRK
250 JMP service
260 EQU% $82
270 EQU% offset MOD 256
280 EQU% 1
290 .title
300 EQU% title$
310 EQU% 0
320 .offset
330 EQU% 0
340 EQU% "(C) XX"
350 EQU% 0
360 .service
370 CMP #9
380 BEQ itshelp
390 RTS
400 .itshelp
410 PHA
420 TYA:PHA
430 TXA:PHA
440 LDA (comline),Y
450 CMP #13
460 BNE more
470 JMP printhelp
480 :
490 .more
500 DEY:STY Ystore
510 LDX #255
520 .check
530 INY
540 INX
550 LDA (comline),Y
560 AND #4DF
570 CMP commands,X
580 BEQ check
590 :
600 LDA commands,X
610 BMI foundit
620 \ moveon

```

VIEW : A Dabhand Guide

```
630 .moveon
640 INX
650 LDA commands,X
660 BPL moveon
670 INX
680 LDY Ystore
690 JMP check
700 :
710 .foundit
720 CMP #$FF
730 BEQ notours
740 STA vector+1
750 INX
760 LDA commands,X
770 STA vector
780 LDY #255
790 .printit
800 INY
810 LDA (vector),Y
820 BEQ zero
830 JSR $FFE3
840 BRA printit
850 .zero
860 LDA #13
870 JSR $FFE3
880 .notours
890 PLA:TAX
900 PLA:TAY
910 PLA
920 RTS
930 :
950 .printhelp
960 LDX #255
970 .helploop
980 INX
990 LDA details,X
1000 BEQ alldone
1010 JSR $FFE3
1020 BNE helploop
1030 .alldone
1050 PLA:TAX
1060 PLA:TAY
1070 PLA
1080 RTS
1090 :
1100 .details
1110 EQU $ 13
1120 EQU $ "VIEW Help ROM"
1130 EQU $ 13
1140 EQU $ "VIEW"
1150 EQU $ 13
```

```

1160 EQUB 0
1170 :
1180 .commands
1190 OPT Fntable("VIEW")
1200 OPT Fntable("STORED")
1210 OPT Fntable("DIRECT")
1220 OPT Fntable("OTHER")
1230 EQUB &FF
1240 :
1250 .stored
1260 EQUB 13
1270 EQU$ "Page Layout:"
1280 EQUB 13
1290 EQU$ "  PL Page Length (66)"
1300 EQUB 13
1310 EQU$ "  TM Top Margin  (4)"
1320 EQUB 13
1330 EQU$ "  BM Bottom Margin (4)"
1340 EQUB 13
1350 EQUB 0
1360 .direct
1370 EQUB 13
1380 EQU$ "Direct Commands:"
1390 EQUB 13
1400 EQU$ "  COUNT <1> <2>"
1410 EQUB 13
1420 EQU$ "  MODE <num>"
1430 EQUB 13
1440 EQUB 0
1450 .other
1460 EQUB 13
1470 EQU$ "Printer Drivers:"
1480 EQUB 13
1490 EQU$ "  FX80  -  Epson FX80"
1500 EQUB 13
1510 EQU$ "  C120  -  Citizen C120D"
1520 EQUB 13
1530 EQUB 0
1540 .view
1541 EQUB 13
1550 EQU$ "VIEW Help ROM"
1560 EQUB 13
1570 EQU$ "  Stored"
1580 EQUB 13
1590 EQU$ "  Direct"
1600 EQUB 13
1610 EQU$ "  Other"
1620 EQUB 13
1630 EQUB 0
1640 }
1650 NEXT

```


VIEW : A Dabhand Guide

```
1660 ENDPROC
1670 :
1680 DEF FNtable($char%)
1690 N%=0:label$=""
1700 REPEAT
1710 upper=char%?N%
1720 upper=upper OR &20
1730 label$=label$+CHR$(upper)
1740 N%=N%+1
1750 UNTIL N%=LEN($char%)
1760 [OPT pass
1770 EQU$ $char%
1780 EQU$ EVAL(label$) DIV 256
1790 EQU$ EVAL(label$) MOD 256
1800 ]
1810 ~pass
```

Listing 18.5. The VIEW Help ROM.

19:A Matter of Style

When writing a document, whether it be a single sheet letter to business clients or a lengthy report to a superior, presentation can be as important as content. Bond quality paper, error free spelling and a letter quality printer with a new ribbon will complement a fine piece of prose, but for full effect, the overall 'look' also has to be right. Trained typists call this 'good display'. Here are a few tips.

Typing primers warn against margin sizes that are inconsistent with, or out of proportion to, the text and stationery. A poor choice can push the text into one corner of the page. They also advise that attention be given to the 'shape' of the text. Scrappy little paragraphs, headings that are not distinct enough from the text, and lonely lines (a single line of text adjacent to a page break) can all mar this shape. Starting a new paragraph or in particular a new section at the very bottom of a page can make it 'disappear' from the reader's eye. Place a conditional page eject command such as PE 5 before the new section begins, to ensure that there is enough of the paragraph on the page to capture the eye.

An expert typist acquires a feel for good display, but with most wordprocessors it is more difficult. Whatever the ads say, what you see is never really what you get. What you see is only half a page at a time!

If you write many documents of the same type, perhaps lots of letters, memos or weekly reports, then it can be useful to define a standard style for each type of document. The edit commands that would normally be put at the top of each type of document should be stored in a file of their own, one file for letters, one for reports and so on. The correct file can be used each time you start a new document, so enter:

LOAD MEMO

to start writing a memorandum with a 'style file' called MEMO. That way, all your memos will have a consistent layout. Such a style file could even contain constant elements of the letter or memo

itself, or define standard macros used later on (perhaps to number report sections). See the example style file in figure 19.1

The style file needs to be loaded in rather than just printed (for example PRINT MEMO MD where the text of the memo is in the file MD). Although printing MEMO in this way would correctly set the margins and define any macros, the text in MD would not be arranged (justified) according to the style ruler in MEMO. View justifies text while it is written, and MD would not know about the ruler in the style file until print time.

Finally proof readings. Always read what you have written - more importantly always read it on paper before you print the final draft. Many people feel that they can edit text 'on screen' with no problems, but speaking as one who has done a lot of editing I know from experience that screen editing is no substitute for paper editing. So my rule is always edit on paper and transfer your changes from paper to VIEW before the final print out.

```

CO style file for memo - RS plc
PL 50
TN 4
BN 4
NN 4
FN 4
LN 15
LS 0
DH ////
DF ////
.. .....*.....C
DE N E N D

LJ From : Bruce Smith
LJ To :
LJ Ref :
LJ Date : 10

LJ Subject

CO memo for 20 lines of text * initial

```

Figure 19.1 Example Style File for a typical office memorandum.

NB. Note use of 1D to automatically include the date: this will only work on a Master 128 with real-time clock.

20:OverView

At the time of going to press Acorn announced the launch of the OverView cartridge for the Master 128. VIEW and ViewSheet are supplied in the main system ROM on the Master 128, the OverView cartridge contains the rest of the VIEW family products namely, ViewStore, ViewSpell, ViewIndex, ViewPlot and the VIEW Printer Driver Generator. Extended *HELP information on the VIEW family products is also included in the OverView cartridge.

In addition to bringing the remaining members of the VIEW family together into the Master 128, OverView allows much greater integration between the packages making data transfer between family members a quicker and easier process.

For example suppose VIEW is being used to prepare a report on some sales figures which are to be calculated in ViewSheet. It would be nice to get the data from ViewSheet and include it as a table in the VIEW written report.

The normal way to do this would be to save the report text, enter ViewSheet and make the calculations. The results could then be *SPOOLED to a new file using SCREEN or alternatively noted/printed on paper. VIEW would be re-entered, the report loaded back in and the spooled file read in or the printed results entered manually, this would involve the following operations at least:

*WORD	(enter report)
SAVE Report	
*SHEET	(enter spreadsheet)
*SPOOL Sheet	
SCREEN	
*SPOOL	
*WORD	
LOAD Report	(set marker 1 in text)
READ sheet 1	(edit spooled text)

All rather long winded.

A new *KEEP command allows the current contents of memory, including all pointers, such as cursor position and the current SETUP options etc to be save to disc or sideways RAM. If the Sideways RAM option is used then the save is virtually instantaneous. ViewSheet could then be entered and its contents saved in the same way. When VIEW is re-entered because everything is saved by *KEEP it was as though you never left and the extra information can be combined as required. Several separate operations are still required but the time taken is drastically reduced.

New Commands

The following two new *commands are provided by the OverView Cartridge and are relevant for use with VIEW:

*KEEP

The command is used in the following ways:

*KEEP ON
*KEEP OFF
*KEEP RAM

As described above this will save the current VIEW context including SETUP options and cursor position. Three KEEP files are saved per language and files for several languages (ie VIEW, ViewSheet, ViewStore) may be kept on disc while two may be kept if the full 64k of sideways RAM is available. In this case the *KEEP RAM command must be used, and attempts to keep a third language status will cause OverView to revert to disc storage.

*WIDE

The command is used in one of two ways:

*WIDE ON
*WIDE OFF

This provides an extra pseudo screen mode in that it allows the screen to display 106 characters instead of 80 characters in Modes 0 and 3 and 53 characters across the screen instead of 40 in Modes 4 and 6 and of course all their shadow equivalents).

It is hoped that full details of OverView will appear in the accompanying volume: **ViewSheet and ViewStore: A Dabhand Guide.**

Appendices

A: Quick Command Guide	196
B: Quick Stored Command Guide	200
C: Quick Highlight Code Guide	203
D: Quick * Command Guide	204
E: Quick Error Guide	207
F: Quick ViewSpell Guide	210
G: Quick ViewIndex Guide	213
H: Priority VIEW	214
I: Increasing your Bytes Free	217
J: Epson Printer Control Codes	219
K: Technical Notes	226
L: VIEW - Version Differences	227
M: The Programs Disc	231
N: Dabhand Guides for your Micro	233

A:Quick Command Guide

The following pages contain a quick summary of VIEW commands that can be used in Command mode. Items bounded by [] are optional. M stands for marker number.

CHANGE <target> <result> [MM]

Change all occurrences of <target> word to <result> word taking into account setting of FOLD. Limit to bounds of markers 1 and 2 if specified.

CHANGE/<phrase one>/<phrase two>/ [MM]

Change all occurrences of <phrase one> to <phrase two> taking into account setting of FOLD. Limit to bounds of markers 1 and 2 if specified.

CLEAR

Remove all set markers from text.

COUNT [MM]

Count the number of words in the document currently held in memory. Limit to bounds of markers 1 and 2 if specified.

EDIT <fileIN> <fileOUT>

Start continuous processing operation, allowing long documents to be written/edited. The input file <fileIN> is read and new text/edited text is written to the output file <fileOUT>. Associated commands are MORE, FINISH and QUIT.

FIELD <number>

Assigns the TAB function to the key whose ASCII number is specified in <number>. FIELD 65 would make 'A' act as the TAB key.

FINISH

Finishes an EDIT session. Text and any text remaining in <fileIN> is written to <fileOUT>. Both <fileIN> and <fileOUT> are then

closed. Memory is cleared. Associated commands are EDIT, MORE and QUIT.

FOLD ON/OFF

Defines whether VIEW distinguishes between character case during a CHANGE, REPLACE or SEARCH. Case is not significant with FOLD ON. Case is significant with FOLD OFF. 0/1 can be substituted for ON/OFF respectively.

LOAD <filename>

Loads the file specified by <filename> into VIEW. Any existing text is erased.

MICROSPACE [<number>]

Allows microspacing providing printer and printer driver support microspacing. The width of characters is given in <number>/120ths of an inch. If the driver supports microspacing an (m) will be displayed after the driver's name.

MODE <number>

Sets the screen mode to <number>.

MORE [M]

Used in an EDIT session to read more text into VIEW. Any text already in memory is written to the output file <fileOUT> and more text is read in from the input file <fileIN>. Marker 1 may be set and specified after the command. In this case text up to the marker will be written to <fileOUT> and more text will be read in to the end of the remaining text. Associated commands are EDIT, FINISH and QUIT.

NAME [<filename>]

Sets the current filename in <filename> to the current editing document. The file name will be displayed in the status information and used with SAVE. In versions of VIEW 3.0 and later only.

NEW

Clears any text from memory and resets pointers for a new document.

PRINT <file1> [<file2> <file3> [<morefiles>]]

Prints out the files specified after the command one by one. The process is continuous and therefore requires continuous stationery in the printer. The process can be stopped at any time by pressing the ESCAPE key.

PRINTER [<filename>]

Loads the printer driver called <filename>. Typing PRINTER will delete any printer driver previously loaded.

QUIT

Aborts an EDIT session: <fileIN> and <fileOUT> are closed immediately with no updating of files. Generally this command should not be used unless you have just started an EDIT session. Associated commands are EDIT, MORE and FINISH.

READ <filename> [M]

Reads in the file named in <filename> to the bottom of current text, unless a marker is specified in which case text is read to the marker. This does not destroy any text already in memory. If there is not enough memory to read in all the file then only as much as can there is room for will be read in. In VIEW 3.0 and greater an error message 'Not all read in' is given. This command allows non VIEW files to be read in, for example *SPOOLED ViewSheet or ViewStore files, or files from other wordprocessing programs.

SAVE [<filename>]

Saves the current document using the filename specified by NAME. If a filename is specified after SAVE then the file will be saved using this filename.

SCREEN [<file1> <file2> [<morefiles>]]

Displays the current document in memory to the command mode screen. All stored commands are acted on so that a 'preview' of the document is possible. If a filename or series of filenames are specified then these will be previewed from the current filing medium, without affecting the file currently in VIEW.

SEARCH <string> [MM]

Locates the first occurrence of <string> with the document in memory. Pressing function key CTRL-F1 will locate the next occurrence of <string> and so on. Limit to bounds of markers 1 and 2 if specified.

SETUP [F1]

Sets text mode flags specified.

F=Formatting

J=Justify mode

I=Insert mode.

Only available in VIEW 3.0 and later.

SHEETS <file1> [<file2> [<morefiles>]]

Works as PRINT except that VIEW will pause at the end of each page until you press a key. This allows single sheet stationery to be utilised. Pressing M will miss the next page, ie, it will not be printed. pressing Q or ESCAPE will cease printing.

WRITE <filename> [MM]

Writes text to the file specified in <filename>. The main advantage of WRITE is when used with markers. Specifying marker numbers will write the text within the markers to <filename>.

B:Quick Stored Command Guide

Items with square brackets, [], are optional.

Text Position

CE text

Centre

Centres the line of text following between current left and right margins.

RJ text

Right Justify

Right justifies the text to the right margin. TABs are reduced to single space characters.

LJ text

Left Justify

Left justifies the text to the left margin.

Headers and Footers

DH /left/centre/right/

Define Header

Defines header to be printed. Text may be in three positions, left, centre or right. If a component is not required the / should still be used but left empty. Thus to define a header with no left component use DH//centre/right/

DF /left/centre/right/

Define Footer

Defines footer to be printed. Text may be in three positions, left, centre or right. If a component is not required the / should still be used but left empty. Thus to define a footer with no left component use DF//centre/right/

HE ON/OFF

Headers

Turns header printing ON or OFF as specified.

FO ON/OFF

Footers

Turns footer printing ON or OFF as specified.

Macros and Registers

DM [nn]

Define Macro

Defines a macro whose two letter (or one letter) name is nn. Letter case is significant, so NN and nn are two different macro names.

EM

End Macro

Ends macro definition.

SR <letter> <number/expression>

Set Register

Sets register <letter> to a value defined by <number> or <expression>. <expression> may take other register letters as an argument.

Page Layout

PB ON/OFF

Page Breaks

Turns page breaks ON or OFF as specified. With PB OFF then text will be printed continuously.

PL <number>

Page Length

Sets page length to <number> lines per page.

TM <number>

Top Margin

Sets the top margin to <number> lines.

HM <number>

Header Margin

Sets the header margin to <number> lines.

FM <number>

Footer Margin

Sets the footer margin to <number> lines.

BM <number>

Bottom Margin

Sets the bottom margin to <number> lines.

LM <number>

Left Margin

Sets the left margin to <number> spaces.

LS <number>

Line Spacing

Sets line spacing to <number>. Thus <number> blank lines will be printed after each printed text line.

TS ON/OFF [<number>]

Two Sided

Reverses left and right components of header and footers on successive pages, catering for two sided documents. If <number> is specified then a margin of <number> spaces is printed on the inside of the document.

PE [<number>]

Page Eject

Ejects the current page and will then either end, if no more text, or carry on printing if there is more text. If a <number> is specified then the page eject becomes a conditional one, with it occurring only if there are fewer than <number> of lines remaining on the page.

OP

Odd Page

Ejects to the next odd numbered page, ie, 3,5,7 etc.

EP

Even Page

Ejects to the next even numbered page, ie, 2,4,6 etc.

HT */- <number>

Highlight

Sets highlight * or highlight - to <number>.

CO <text>


Comment


Places a comment in the text which will not be printed. Actually this is not a real command. Typing any two letter which are not defined as a macro and are not one of the other stored commands will have the same effect. ie.

 NB This is a draft version
would not be printed.

C:Quick Highlight Guide

Standard Highlight Codes

Underlined text - highlight 1 SHIFT-f4 shown as 



Bold text - highlight 2 SHIFT-f5 shown as 

Extended Highlight Codes

If you have an Acornsoft Printer Driver Generator then you can access the extended highlight set. The extended set is brought into operation using the stored command:

HT 2 130

The table below lists the highlight sequences available.

 represents a highlight 1 character (SHIFT-f4) and  a highlight 2 character (SHIFT-f5).

Highlight Effect



Reset printer

Switch underlining on and off

Switch bold on and off

Begin subscripting

Begin superscripting

Revert to normal print after super- or subscripting



Select alternative font (Pica) on or off

Switch italics on or off

The default highlight characters are reselected with the stored command:

HT 2 129

D:Quick * Command Guide

The following * commands are often of use. A list of them, with brief descriptions, follows. See the relevant chapter in this book or the appropriate manual for full details.

General Commands

- *BASIC Enter BASIC programming language.
- *SHEET Select ViewSheet spreadsheet.
- *SPELL Select ViewSpell spelling checker.
- *STORE Select ViewStore database.
- *WORD Select VIEW wordprocessor.

Safe Filing System Commands

The following commands will not affect any document in VIEW when used.

- *ADFS Select Advanced Disc Filing System
- *DISC Select Disc Filing System
- *NET Select Net Filing System
- *TAPE Select Cassette Filing System
- *ACCESS Locks or unlocks files to prevent overwriting
DFS: *ACCESS <filename> L
ADFS/NET: *ACCESS <filename> WRL
- *BACK Reselect last directory selected in ADFS
- *BUILD Makes a text file on disc
- *CAT Catalogues files on disc/in current directory
- *CDIR Creates named directory on ADFS or NET
*CDIR Letters : makes directory called Letters
- *CLOSE Closes any open files on disc
- *DELETE Deletes the specified filename - use with care
*DELETE Chap1 : deletes the file called Chap1
- *DESTROY Destroys the contents of a specified directory
*DESTROY Letters : will list files in ADFS directory

called letters. Typing YES will then cause all files to be erased. In DFS *DESTROY L.* will similarly list files in directory L and typing Y will erase them.

- *DIR Select specified directory
*DIR Letters : will select the Letters directory
- *DISMOUNT Has the reverse action to *MOUNT - closes all open files
- *DRIVE Select disc drive 0,1,2, or 3 as specified
- *DUMP Dump specified file to screen
- *EX Extended catalogue
- *FREE Print bytes used and free on ADFS, NET and later versions of DFS
- *FORMAT Format new disc
- *INFO Provides information about specified file(s)
- *LCAT Catalogue library directory
- *LIB Set the library directory to that specified
- *MAP Show position of file blocks on disc
- *MOUNT Mount newly inserted ADFS disc
- *OPT *OPT 4,3 sets the EXEC option
*OPT 1,1 shows file details on save/load
- *PRINT As *TYPE on a Master 128/Compact. *TYPE on these machines behaves slightly differently.
- *RENAME Rename existing file: *RENAME <current> <new>
- *SAVE Save block of memory specified using name specified
- *SRLOAD Load ROM image into RAM bank (BBC B+, Master 128 and Master Compact). Do NOT use Q option
- *SRSAVE Save RAM bank contents as specified. Do not use Q option
- *TITLE Name disc as specified: *TITLE Book
- *TYPE Types named file to screen
- *VERIFY Verifies formatted disc

Unsafe Filing System Commands

The following commands will/can destroy any document in VIEW if used. Always save text before use.

- *BACKUP Copies contents of one disc to another
- *EXEC Execute contents of named file
- *COMPACT Close up any gaps on disc to free more space
- *COPY Copy named file to another disc/directory
- *SRLOAD Load ROM Image into RAM bank when used with Q option
- *SRSAVE Save contents of named RAM bank when using Q option

*FX Commands

Further details on the *FX commands below can be found in Chapter 14.

- *FX 5 Select parallel printer output
- *FX 5,2 Select serial printer output
- *FX 6 Enable software linefeeds
- *FX 6,10 Disable software linefeeds
- *FX 8 Select serial printer transmission rate
- *FX 11 Set key auto repeat rate
- *FX 12 Set key repeat delay
- *FX 202,48 Turn CAPS LOCK light off
- *FX 228,1 Enable function keys for use in Edit Mode

*TV Commands

Further details of *TV commands can be found in Chapter 14.

- *TV 0,1 Turn interlace off
- *TV 255,1 Move display down screen

Appendix E: Quick Error Guide

At times you will request VIEW to carry out a certain task and it will simply return with a message. This error message should give you a brief indication as to the problem. What follows is a more detailed description of each error message.

Bad file

This error is given after you have requested a printer driver to be loaded. It is issued because VIEW does not recognise the file you have specified as a printer driver program. The most likely answer is that you have tried to load a driver which is in fact a VIEW text file or perhaps a printer driver that for another program. The first is easy to check. If the file is a driver then check it using your VIEW printer driver generator program.

Bad filename

VIEW reports this when you have used a filename that is longer than 19 characters in length or you have used illegal characters in a LOAD, SAVE, READ or WRITE operation. For example:

```
LOAD P.
```

would report this message due to the full stop. Filenames in tape and disc filing systems may only be 10 or 7 characters long respectively so the length restriction should not generally apply in the first instance. In ADFS root names can total longer than 19 characters so to avoid this error simply move nearer the final directory. If the name is alright by VIEW but the filing system doesn't like it - ie a 9 letter name in DFS, then the filing system will generate a similar message - usually 'Bad name'. You can't tell just by looking which bit of the system generated the error.

Bad flag

Only applies to VIEW versions 3.0 or greater. It is reported when you have used an illegal parameter as part of the SETUP command. Only flags F, J and I may be used. Typing:

```
SETUP X
```

would create this error as there is no X flag.

Bad marker

This error is generated when you refer to a marker other than 1,2,3,4,5, and/or 6, for example COUNT 8 9. Only markers 1 to 6 are allowed.

Bad mode

You have used the MODE command but specified a non-existent mode. For example MODE 8. Only Modes 0 to 7 and, on micros with shadow memory, 128 to 135 are permissible.

Printer does not support microspacing

You have entered the command MICROSPACE but the printer driver currently loaded does not support microspacing. The answer is to load a printer driver that will support microspacing.

File not found

You have tried to LOAD or READ a file that does not exist in the current directory. Is it in another directory or disc - or have you misspelt the filename? With LOAD the message is sometimes just 'Not found'.

Marker not set

This error is given because you have made a reference to a marker that has not been set. Check you have used the correct marker numbers and/or set the marker again.

Mistake

You have entered a command into VIEW that it does not understand. Check spelling and typing of command.

Nested macro call

This error tells you that you are trying to call a macro from within a macro: this is simply not possible. Re-write macro definition(s) or delete nested macro call.

Not all read in

Only applies to VIEW versions 3.0 and greater. This message is given when you have asked for a file to be READ in. Only as much of the

text as there is space for will be read in and if any text is not read in then this message will be displayed.

Not enough memory

The file you are trying to load is too big and will not fit into the available memory. See Appendix I for details on possible means to circumvent this.

No string found

The target string you have specified in a CHANGE, SEARCH or REPLACE command has not been found.

No target given

You have not specified a target string for a CHANGE, SEARCH or REPLACE command.

No text

You have tried to load a non-VIEW file. Type NEW and then try to READ the file into VIEW.

F:Quick ViewSpell Guide

Commands

Items within square brackets, [], are optional.

***SPELL**

Select VIEWSpell.

ADD [<dictionary>]

Allows current list of words to be added to the current dictionary unless a <dictionary> is specified.

AW <filename>

Selects Add Word mode and allows words to be added to the User dictionary specified by <filename>.

CHECK

Checks the current list of words against the Master dictionary and then, should you so wish, a User dictionary.

CMARK

Function as MARK but allows marked file to be written to a second disc. Destination disc and source disc prompts are displayed by ViewSpell.

CONTEXT [<word>]

Displays unrecognised words in their context, by printing the line in which they are contained. May be limited to a specific <word> if specified.

CREATE <filename>

Create a blank User dictionary called <filename>.

DW <filename>

Selects Delete Word mode and allows words to be deleted from the user dictionary specified by <filename>.

LIST

After a LOAD operation, LIST will list the unique words in the document. After CHECK it will list the unrecognised words.

LOAD <filename>

Load the named file.

MARK <filename>

Marks all unrecognised words in the current document and saves the marked file using the name specified in <filename>. See CMARK.

MARKER [1 2] [<character>]

Sets Marker 1 and/or Marker 2 to be the specified <character>.

MODE <number>

Select the <number>ed screen mode.

NAME [<filename>]

Sets the current filename if specified and will be used by SAVE.

NEW

Clears the contents of memory in readiness for a new document.

PREFIX [TMU]

Defines the prefix that will be added to the T(ext) M(aster) and U(ser) loading operations.

READ <filename>

Reads in the document specified in <filename> and adds the new list of unique words to those already present.

SAVE [<filename>]

Unless specified SAVE will save the list of unique or unrecognised words to the current NAMED file

SCREEN <filename>

Displays the file specified by <filename> to the screen.

SEARCH <word> [<dictionary>]

Searches the master or specified <dictionary> for <word>. <word> may include the ? or * wildcards if correct spelling is unknown.

USER <dictionary>

Sets the User dictionary to <dictionary>.

G:Quick ViewIndex Guide

Files

LINDEX	- Intermediate Index file
INDEX	- PRINTER driver for index creation
SHIFT-BREAK	- to select ViewIndex sort program
O.INDEX	- final Index file to load into VIEW

Templates

TPP : Page numbering template
 TPS : Section numbering template

Section numbering

The following macros should be used to identify a chapter, section or subsection:

IC Chapter
 IS Section
 IT Subsection

Creating an Index

- * Mark all words requiring indexing
- * PRINTER INDEX (Load index printer driver)
- * PRINT the index using either TPP or TPS macro first:

```
PRINT TPP <file1> <file2> etc
PRINT TPS <file1> <file2>
```

- * Press SHIFT-BREAK to start index sort program

H:Priority VIEW

If you use your micro almost entirely for VIEW wordprocessing then it is worth configuring your system so that VIEW is always selected by default when you turn your micro on. The following notes should help you do this with your machine. In some cases it will be necessary to change the position of the VIEW ROM - if you are uneasy about this, your local dealer or computer club enthusiast will undertake it for you. Do not attempt to remove the lid of your micro without unplugging the micro from the mains electricity first.

BBC Model B

To configure VIEW as the default language on a BBC B micro it is necessary to fit the VIEW ROM itself into ROM socket 15. With the lid removed and the keyboard facing you, the four ROM sockets can be found just above the far right hand side of the keyboard. Socket 15 is the rightmost of the four, as shown in figure H.1. If the socket is already occupied then the ROM it contains must be removed and inserted into another empty socket.

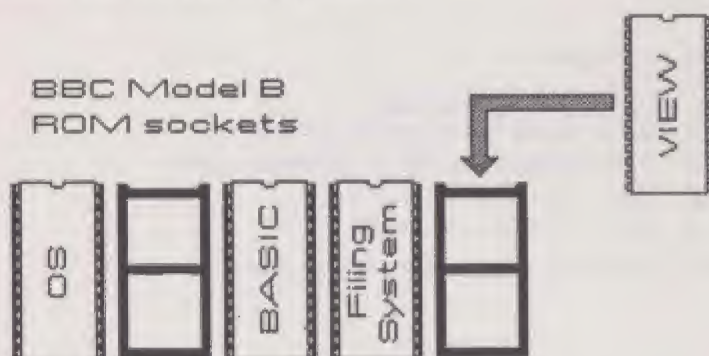


Figure H.1 - Model B ROM layout

BBC Model B+

With the lid removed, the six B+ ROM sockets can be seen to the back left-hand side, to the right of the keyboard - they will be arranged in two banks of three ROM slots. The VIEW ROM chip must be fitted in the top left socket of the six, next to the DFS ROM, as

shown in figure H.2. To the right of the lower bank of there there is a small link, labelled E15. This should be moved so that it sits across the centre and northernmost pin.

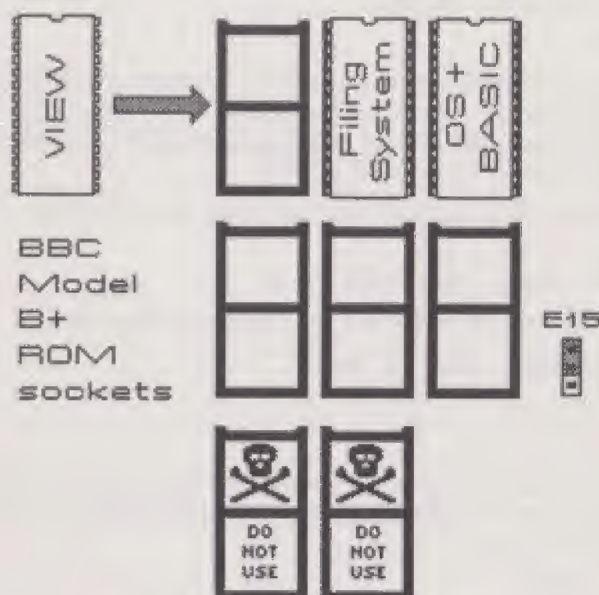


Figure H.2 - BBC B+ ROM layout

Master 128

VIEW can be set as the default language by entering the following command:

```
*CONFIGURE LANG 14
```

and then pressing the CTRL-BREAK keys together. VIEW will now be selected when the Master is switched on.

Master Compact

VIEW is supplied on the Welcome disc as a ROM image that is loaded into sideways RAM when required - it is not possible therefore to have VIEW selected as soon as you switch the Compact

on. However, we can make a VIEW Boot disc onto which we can also place VIEW files.

To do this first format a new disc - switch the Compact on, place a new disc in the drive and type:

```
*FORMAT 0 L
```

and then answer YES in response to the Compact's question. The disc will be formatted and verified. Label this 'VIEW Disc'. Load the Welcome disc and boot this by holding down SHIFT-BREAK. Select the Applications Menu and then VIEW.

Replace the newly formatted disc and type the following:

```
*MOUNT  
*SRSAVE WORD 8000+4000 ? 0
```

Now enter edit mode and write the following !Boot file:

```
*SALOAD WORD 8000 ? 0!  
*WORD  
MODE 131  
NEW
```

add to the end of this any other items you may wish included as part of your !Boot file. Save the file to disc using the WRITE command and set *OPT 4,3 thus:

```
WRITE !Boot  
*OPT 4,3
```

Now when you switch on simply insert and boot the VIEW Disc.

I:Increasing the Bytes Free

Each character you enter into VIEW is stored in a single byte of the micro's memory. The status information on the Command mode screen shows the number of bytes free, which can be directly related to the number of characters that can be entered into VIEW. Of course, therefore, for longer documents you will find that at some stage you will run out of bytes and VIEW will give you a Memory Full error and not allow you to enter any more text. There are several options open to gain more bytes free, depending on the micro you are using. In all cases the software option is to save your current work and then use the EDIT command to begin a continuous processing session. Refer to Chapter 13 for more details.

BBC Model B

If you do not wish to pursue the EDIT option, you must change to a lower resolution screen mode. Unlike the B+, Master and Compact the standard BBC B does not have shadow memory fitted, thus the normal memory that would be used to store VIEW text must be used to display the screen. A simple and effective upgrade is to add a Shadow RAM board to your BBC B. There are several varieties available, so examine the magazine advertisements and consult the many reviews that have been carried in the Acorn associated magazines.

BBC Model B+, Master 128 and Master Compact

Ensure that you are using a shadow screen mode, thereby releasing the normal memory for VIEW text. This can be done with a simple command entered of course from Command mode:

```
MODE 131
```

If shadow RAM is in use then there is very little else you can do other than to use one of the hardware additions outlined below.

Second/Co-processors

A second or co-processor effectively adds an extra 64k of memory to your computer of which about 48k (48,000 bytes) are available for use with VIEW. A 6502 second processor may be fitted externally via a small length of cable to a BBC B, B+ or Master 128. A Turbo Co-processor is a Master 128 product only and fits inside the case of the

computer, although there are products available which allow Master co-processors to be plugged into Model Bs. Neither can be used with a Master Compact.

If you are using VIEW 2.1 with a second processor then the bytes free count will be somewhat less than this. The full second processor memory can be used by purchasing a copy of Acornsoft's Hi-VIEW or upgrading to VIEW version 3.0 or greater.

The 6502 second processor and Turbo co-processor are both Acorn products. However, third party second processors do exist and these can be a cheaper alternative. As always check out the adverts and magazine reviews.

J:Epson Printer Control Codes

The following pages contain control code sequences for the three popular types of Epson printer, including the FX printer codes which form the unofficial printer standard code set. So the FX codes should also be the same as those on your printer if it is sold as being Epson-compatible, for example the Kaga Taxan. If a code does not seem to function as detailed then do consult your own printer manual. Although many printers are sold as being Epson-compatible there are often minor differences.

The control codes are arranged in alphabetical order. The codes are give in decimal numeric form - most begin with 27, which is the ESC code. Most printer driver generators will allow you to enter the sequences using decimal numbers as given, but some will require ESC to be pressed or typed in place of code 27.

Occasionally you will need to supply your own information and this is shown by a series of dots, ie,

27,94,.....

Where various parameters are supplied in the description, the appropriate one should be inserted where specified by a variable within the code, ie,

27,67,n

(Sending 27,67,60 would set the form length to 60 lines). If you wish to use the codes from BASIC then do remember to turn the printer on with VDU 2 and then send a 1 before each code before disabling the printer with VDU 3. ie to send code 27,69 type:

VDU 2,1,27,1,69,3

Chapter 11 contains more information on printers and printer control codes.

FX Printer Codes

Description	Code(s)	Detail
Backspace	8	backspace one place
Bell	7	sounds bell
Bit image set (8)	27,m,n,o	selects various 8 bit graphics modes

VIEW : A Dabhand Guide

Bit image set (9)	27,94,....	selects various 9 bit graphics modes
MSB to 0	27,61	sets msb of following 8-bit data to 0
MSB to 1	27,62	sets msb of following 8-bit data to 1
MSB cancel	27,3 5	cancels above codes
normal density	27,75,....	change to bit image mode
dual density	27,76,....	change to bit image mode
dd double speed	27,89,....	faster and no adjacent dots
quad density	27,90,....	as above but darker
Cancel	24	deletes previous data in print buffer for same line
Carriage return	13	carriage return
Condensed mode on	15	subsequent data printed condensed
on	27,15	alternative for above
off	18	cancels above
Control code select	27,73,n	n=1 or 49 selects codes 0-31 as printable. n=0 or 48 selects as control codes
Delete	127	deletes character (in print buffer)
Double strike set	27,71	sets double strike mode
cancel	27,72	cancels above
Download def'n.	27,38,....	defines download characters
select	27,37,1,0	selects previously defined set
cancel	27,37,0,0	selects ROM character set
ROM copy	27,58,0,0,0	copies ROM character set to download character set
Elite mode set	27,77	following data printed in elite size
cancel	27,80	cancels above i.e. returns to normal print
Emphasized mode set	27,69	all following data printed in emphasized mode
cancel	27,70	cancels above
End of paper on	27,57	selects end of paper detector
off	27,56	deselects end of paper detector
Enlarged mode set	14	enlarged for one line
set	27,14	as above

set/reset	27,87,n	n=1 or 49 all following data printed enlarged. n=0 or 48 cancels
cancel	20	cancels that set by 14
Expansion on	27,54	codes 128-159 & 255 are set as printable, see download
off	27,55	cancels above
Form feed	12	executes form feed
length lines	27,67,n	sets form length as n lines
length inches	27,67,0,n	sets form length as n inches
Half speed	27,115,n	n=1 or 49 sets half speed print. n=0 or 48 cancels
Incremental print	27,105,n	n=1 or 49 sets print and view. n=0 or 48 cancels
Indent	27,108,n	sets n character left margin
Initialise	27,64	initialises printer, including clearing buffer
International set	27,82,n	prints following data from n character set
Italics on	27,52	prints all following data in italics
off	27,53	cancels above
Line feed forward	10	executes line feed
reverse	27,106,n	executes n/216" reverse feed
Margin set	27,108,n	sets n character left margin
Mode select	27,33,n	selects one of 63 type faces
On	17	enables printer
Off	19	disables printer
Page width	27,81,n	sets page width to n characters
Paper feed	27,74,n	executes an n/216" paper feed (0<=n<=255)
Proportional spacing	27,112,n	n=1 or 49 sets proportional spacing. n=0 or 48 cancels
Re-set	27,64	initialises printer, including clearing buffer
Reverse feed	27,106,n	executes n/216" reverse feed
Skip over perforation	27,78,n	skips n lines at page bottom
cancel	27,79	cancels above
Slow speed	27,115,n	n=1 or 49 sets half speed print. n=0 or 48 cancels
Spacing 1/8"	27,48	sets line spacing to 1/8"
7/72"	27,49	sets line spacing to 7/72"

1/6"	27,50	sets line spacing to 1/6"
n/216"	27,51,n	sets line spacing to n/216"
n/72"	27,65,n	sets line spacing to n/72". i.e. dots width
Subscript set	27,83,n	n=1 or 49 sets subscript mode
cancel	27,84	cancels above
Superscript set	27,83,n	n=0 or 48 sets superscript
cancel	27,84	cancels above
Tab horizontal	9	executes horizontal tab
horizontal	27,68,....	sets horizontal tabs
vertical	11	executes vertical tab
vertical set	27,98,....	sets 8 channels of vertical tabs
vertical	27,47,n	executes n th format vert.tabs
		e.g.one format for each of seven pages
vertical	27,66,....	defines vertical tab positions
Underline	27,45,n	n=1 or 49 selects underline, n=0 or 48 deselects underline
Unidirectional print	27,60	prints from left to right for single line
	27,85,n	n=1 or 49 sets unidirectional print. n=0 or 48 sets bidirectional print

Epson MX80 Type 3 Codes

Description	Code(s)	Detail
Backspace	8	backspace one place
Bell	7	sounds bell
Bit image		
normal density	27,75,....	following data printed as bit images
dual density	27,76,....	following data printed as bit images
Carriage return	13	carriage return
Condensed mode on	15	stored and subsequent data printed condensed
off	18	cancels above
Double strike set	27,71	sets double strike mode
cancel	27,72	cancels above
Emphasized mode set	27,69	all following data printed in emphasized mode
cancel	27,70	cancels above

End of paper on	27,57	selects end of paper detector
off	27,56	deselects end of paper detector
Enlarged mode set	14	enlarged for one line
set	27,87,n	n=1 all following data printed enlarged, n=0 cancels
cancel	20	cancels that set by 14
Form feed		12 executes form feed
length lines	27,67,n	sets form length as n lines
length inches	27,67,0,n	sets form length as n inches
Initialise	27,64	initialises printer
International set	27,82,n	prints following data from n character set
Line feed forward	10	executes line feed
Page width	27,81,n	sets page width to n characters
Paper feed	27,74,n	executes an n/216" paper feed (0<=n<=255)
Skip-over perforation	27,78,n	skips n lines at page bottom
cancel	27,79	cancels above
Spacing 1/8"	27,48	sets line spacing to 1/8"
7/72"	27,49	sets line spacing to 7/72"
1/6"	27,50	sets line spacing to 1/6"
n/216"	27,51,n	sets line spacing to n/216"
n/72"	27,65,n	sets line spacing to n/72" i.e. dots width
Subscript set	27,83,1	sets subscript mode
cancel	27,84	cancels above
Superscript set	27,83,0	sets superscript
cancel	27,84	cancels above
Tab horizontal	9	executes horizontal tab
horizontal	27,68,....	sets horizontal tabs
vertical	11	executes vertical tab
vertical	27,66,....	defines vertical tab positions
Underline	27,45,n	n=1 selects underline, n=0 deselects underline
Unidirectional print	27,85,n	n=1 sets unidirectional print, n=0 sets bidirectional print

Epson RX80 Codes

Description	Code(s)	Detail
Backspace	8	backspace one place
Bell	7	sounds bell
Bit image		

normal density	27,75,....	following data printed as bit images
dual density	27,76,....	following data printed as bit images
dd double speed	27,89,....	as above but faster and no adjacent dots
quad density	27,90,....	as above but darker
Carriage return	13	carriage return
Condensed mode on	15	stored and subsequent data printed condensed
on	27,15	as above
off	18	cancels above
Delete	127	deletes previous char. in print buffer
Double strike set	27,71	sets double strike mode
cancel	27,72	cancels above
Elite mode set	27,77	following data printed in elite size
cancel	27,80	cancels above i.e. returns to normal print
Emphasized mode set	27,69	all following data printed in emphasized mode
cancel	27,70	cancels above
End of paper on	27,57	selects end of paper detector
off	27,56	deselects end of paper detector
Enlarged mode set	14	enlarged for one line
set	27,14	as above
set	27,87,n	n=1 or 49 all following data printed enlarged. n=0 or 48 cancels
cancel	20	cancels that set by 14
Form feed	12	executes form feed
length lines	27,67,n	sets form length as n lines
length inches	27,67,0,n	sets form length as n inches
Half speed	27,115,n	n=1 or 49 sets half speed print. n=0 or 48 cancels
Indent	27,108,n	sets n character left margin
Initialise	27,64	initialises printer
International set	27,82,n	prints following data from n character set
Italics on	27,52	prints all following data in italics

off	27,53	cancels above
Line feed forward	10	executes line feed
Margin set	27,108,n	sets n character left margin
Page width	27,81,n	sets page width to n characters
Paper feed	27,74,n	executes an n/216" paper feed (0<=n<=255)
Re-set	27,64	initialises printer, including clearing buffer
Skip over perforation	27,78,n	skips n lines at page bottom
cancel	27,79	cancels above
Slow speed	27,115,n	n=1 or 49 sets half speed print. n=0 or 48 cancels
Spacing 1/8"	27,48	sets line spacing to 1/8"
7/72"	27,49	sets line spacing to 7/72"
1/6"	27,50	sets line spacing to 1/6"
n/216"	27,51,n	sets line spacing to n/216"
n/72"	27,65,n	sets line spacing to n/72" i.e. dots width
Subscript set	27,83,n	n=1 or 49 sets subscript mode
cancel	27,84	cancels above
Superscript set	27,83,n	n=0 or 48 sets superscript
cancel	27,84	cancels above
Tab horizontal	9	executes horizontal tab
vertical	11	executes vertical tab
Underline	27,45,n	n=1 or 49 selects underline. n=0 or 48 deselects underline
Unidirectional print	27,60	prints from left to right for single line
	27,85,n	n=1 or 49 sets unidirectional print. n=0 or 48 sets bidirectional print

K:Technical Notes

Internally, VIEW uses a memory byte for each character letter. The ASCII code of the character itself is stored in memory. Text storage begins at OSHWM+&100, so on a Master where the OSHWM (usually the same as PAGE in BASIC) is set at &E00 the first character in the VIEW document will begin at &F00.

Stored commands are preceded by an &80 byte, the stored command itself being represented in true ASCII format.

Spaces are represented in ASCII, ie, &20. When VIEW formats text it inserts soft spaces. As it may need to remove these if justification is tuned off and the text reformatted it uses &1A to represent hard spaces, though these look to be normal spaces on screen.

The highlight 1 and highlight 2 codes are represented as &1C and &1D respectively in memory. On encountering these codes VIEW will send 128 and 129 to the printer driver respectively.

VIEW does not use an end of file marker to mark the end of the current document in memory, instead it keeps a vectored address at &D and &E (on the language processor) which points to the last character entered in memory.

A consistency byte (&AA) is stored at OSHWM.

L:VIEW Differences

Outlined briefly over the next couple of pages are some of the changes that have been made to improve the operation of VIEW since releases 1.4 and 2.1. When the term VIEW 3 is used this should be taken to mean version 3.0 and greater unless otherwise specified.

General

The version of VIEW supplied with the Master Compact allows control characters to be entered from the keyboard direct. VIEW 3 and later will perform an automatic NEW command when it is selected. Earlier versions did not and could not be used until NEW was issued. On versions 1.4 and 2.1 the NAME and SETUP commands are not available, nor is the PB stored command. Spaces before a command will cause an error - spaces are permissible in VIEW 3 and later.

VIEW 2.1: This has extended *HELP options.

VIEW 3: The OLD command is performed automatically. In versions 2.1 and 1.4 it should be used to regain text after the BREAK key has been pressed. Files are not closed when the automatic OLD is performed. If VIEW needs to split a line because it is too long to fit on the line then the nearest word boundary is used. If there is no word boundary then the 132nd character position is used. The PB command has been added to VIEW 3. The FOLD, TS, PB, HE and FO will now take ON and OFF as arguments in addition to 0 and 1. There are no *HELP options.

Chapter 2

In VIEW 3 and later the COUNT command also counts words in the CE, RJ and LJ stored commands. This was not the case in the earlier versions.

When using on a standard BBC B or B+ pressing the BREAK key with VIEW 3 installed will put you back into MODE 7 and switch the CAPS LOCK back on. Screen mode can be preserved if you make correct the keyboard links.

In VIEW 1.4 and 2.1 the CTRL arrow keys could not be used to move the cursor to top, bottom, beginning and end of line.

Chapter 3

VIEW 1.4: When the SCREEN command is used to preview a file from disc the cursor can sometimes get stuck at the bottom of the screen with the result that the screen display will become locked in effect VIEW remains in Paged mode which makes it impossible to move down through a document in the normal manner when in Edit mode. The way around this is to switch Paged mode off by ESCAPEing into Command mode and pressing CTRL-O.

Function key strips for earlier versions say FORMAT BLOCK for f0, not FORMAT PARAGRAPH.

Chapter 4

VIEW 2.1/1.4: EDIT and LOAD were not trapped as an error when used with the cassette filing system (CFS) selected. In VIEW 3 the 'Not with cassette' error message is given.

Chapter 5

In VIEW 2.1 and 1.4 the READ command reserved memory, now it does not though it requires about 150 bytes of memory to be free so that it can work. In VIEW 3 and later the message 'Not all read in' is given should there not be enough memory for all the text to be read in. No message is given in earlier versions of VIEW.

In VIEW 3 the DELETE UP TO CHARACTER action has been revised. Now Space will only find spaces, TABs find TABs. Pressing Highlight 1 or 2 will find the corresponding highlight.

In VIEW 1.4 and 2.1 the CLEAR command only erased markers 1 and 2 from text - markers 3 to 6 were left in position and had to be deleted 'manually'.

Chapter 6

VIEW 1.4: If you find words are becoming concatenated when you format text this is because you have deleted the soft spaces. VIEW 1.4 inserts after hard spaces (normal spaces) to justify text.

Older function key strips have INSERT MODE for CTRL-I4, not INSERT/OVERTYPE. VIEW 3 always regards the end of a line as a new word with SHIFT-right or SHIFT-left, so that if you're at the start of the last word on a line, SHIFT-right takes you to the end of

the line on version 3.0, but to the first word of the next line in version 2.1.

In VIEW 2.1 and 1.4 Format Block operations did not stop on lines with left margins TABs followed by spaces and TABs.

Chapter 7

VIEW 2.1/1.4: When SEARCH/REPLACE/CHANGE are used the stored command field is also searched. It is not in VIEW 3 and later. In VIEW 2.1/1.4 the WILD command can be used to specify or change the wild character in one of these operations. The normal wild characters are ?~and !. These can be changed by specifying the character after the WILD command. For example WILD \$ would change the ? wild character into \$.

Chapter 11

VIEW 1.4: can only PRINT files from a disc rather than from disc and/or memory. Compact VIEW: The default printer driver will allow characters which have the top bit (bit 7) set to pass through the driver and be printed. Highlight codes in VIEW 3 are shown in inverse video in modes 0-6. On earlier versions they are not and appear as * and -. The HT command will now accept the underline or asterisk characters as arguments as well as 1 and 2: HT * 130 sets highlight 2 to 130.

Chapter 13

VIEW 2.1 and 1.4: Use of QUIT and FINISH will close all open files and not just the ones currently in use.

Appendix I

VIEW 2.1: This version will not work on a BBC B+ in Shadow mode without a special machine code patch program being run first of all. This is available from Acorn, and was published in the July 1985 issue of Acorn User (page 112).

Command Changes

The three major releases of VIEW have seen some commands come and go. The table below lists the commands and abbreviations of each for each release of VIEW. A dash indicates that there is no

shorter form of the command and n/a that the command is not available.

Command	VIEW 1.4	VIEW 2.1	VIEW 3
CHANGE	C	C	C
CLEAR	-	-	CL
COUNT	-	-	CO
EDIT	-	-	ED
FIELD	-	-	FI
FINISH	F	F	F
FOLD	n/a	-	FO
FORMAT	-	-	FOR
LOAD	-	L	L
MICROSPACE	-	-	MI
MODE	-	-	M
MORE	M	M	MO
NAME	n/a	n/a	N
NEW	-	-	-
OLD	-	-	-
PRINT	P	P	P
PRINTER	-	-	PRINTE
QUIT	-	-	-
READ	-	-	RE
REPLACE	R	R	R
SAVE	-	-	SA
SCREEN	-	-	SC
SEARCH	S	S	S
SETUP	n/a	n/a	SET
SHEETS	-	-	SH
WILD	-	-	n/a
WRITE	-	-	W

M:The Programs Disc

A disc of programs is available from Dabs Press containing all the programs listed in this book plus several others, including a full feature Printer Driver Generator and a VIEW Spooler program ideal for preparing text to transfer into other packages or via a modem down the telephone lines. This is what you get:

- * The Dabs Printer Driver Generator
- * The Printer Driver Generator Manual
- * A series of ready made printer drivers
- * VIEW Spooler
- * VIEW Manager
- * Extended Catalogue program
- * An extended VIEW Help ROM image for sideways RAM
- * The Screen Layout driver
- * The VIEW recovery program
- * VIEW File Checker
- * Highlight Checker
- * The DIY Printer Driver Generator
- * A series of !Boot Files
- * A dictionary of Words for ViewSpell extracted from this book

The Dabs Printer Driver Program: Written in BASIC, this program will allow you to create a versatile printer driver for your own printer with a maximum of 31 highlight codes possible depending on the types of codes required by your printer. The printer driver is made easy to use as it does not require typing of complicated and difficult to remember highlight sequences. For example if you wished to use enlarged print on you printer and this had been defined as highlight code 134 then you simply set highlight 1 or 2 to this accordingly with a stored command, ie,

HT 2 134

Use of highlight 1 will now enable enlarged print. Please note that the driver does not support the use of microspacing. Full instructions on using the driver are provided in VIEW format ready for you to read from VIEW or print out. In addition, the Programs Disc also contains ready to use printer drivers for a variety of printers including the Epson FX80, Kaga Taxan, and Citizen C120D.

The VIEW Spooler: Written by Graham Bell, this program will allow you to spool true ASCII text, which makes it ideal for

preparing text to be transferred into other wordprocessors or to transmit down a telephone line via a modem. The program is used like a printer driver and spools text to a file on disc plus lots more!

Of course the disc contains all the programs from this book, including an extended VIEW Help ROM image, VIEW Manager, a variety of !Boot files plus a small User dictionary of unique words compiled from the text in this boook and not present in the standard Master dictionary.

The disc is available for the BBC B, B+ and Master 128 in 40 track 5.25in DFS format for just £7.95. A copy program will allow you to convert the disc for use with an 80 track drive. The software can also be transferred to Econet or a hard disc. It is also available in 3.5in ADFS format for Master 128 and Master Compact for £9.95. The disc is not copy-protected, and comes in a handsome illustrated plastic wallet with full instructions. The price is *inclusive* of VAT and postage and packing.

To obtain your copy of the disc then use the order form below, or a copy of it, and send it with a cheque or postal order or sterling money order to Dabs Press at the address on page 234. Government, education, and companies may send an official order. Dealer enquiries are welcome.

-----✂-----

Please rush me a copy of the Programs Disc to accompany Bruce Smith's Dabhand Guide to VIEW. I require the following version (please delete as required):

5.25in DFS (£7.95) / 3.5in ADFS (£9.95)

Name

Address

..... Postcode.....

I enclose a cheque/PO for no.....

Tick here if you require a VAT receipt

N:The Dabhand Guide Guide

The following Dabhand Guides covering the BBC and Master series micros are planned for 1987. Publication dates and contents may be subject to change. All quoted prices are inclusive of VAT (on software - books are zero-rated), and postage and packing.

ViewSheet and Viewstore: A Dabhand Guide by Graham Bell

Publication : July 1987. 250 pages

Book: £12.95. Disc: DFS 5.25in £7.95 ADFS 3.5in £9.95. Book and disc together £17.95 (ADFS £19.95)

Written by Acorn User's Graham Bell this is the first complete guide for ViewSheet, ViewStore and ViewPlot. Divided into four sections, this book serves as an introduction to these members of the VIEW family whilst providing an indispensable reference guide to more advanced users. Worked examples are provided throughout, and numerous utility programs are included. The perfect companion to VIEW: A Dabhand Guide!

VIEW Family Applications: A Dabhand Guide by Bruce Smith and Graham Bell

Publication: October 1987. Software with manual
£14.95 for the package.

Bruce Smith and Graham Bell team up to provide a whole range of expert utilities for users of any VIEW products. Includes automatic file backup, front ends to the VIEW family, sideways RAM software, how to write a ViewStore utility, advanced printer and screen drivers plus many more.

Master Operating System: A Dabhand Guide by David Atherton

ISBN 0-870336-01-1

Publication : May 1987. 250 pages

Book: £12.95. Disc: DFS 5.25in £7.95 ADFS 3.5in £9.95. Book and disc together £17.95 (ADFS £19.95)

What all BBC B+, Master 128 and Master Compact owners have been itching to get - BBC expert David Atherton's Guide to the MOS

covering all aspects of programming at OS level. Areas covered include, *commands, the 65C12 microprocessor, OSBYTE/OSWORD calls, shadow RAM, sideways RAM, the hardware, differences between micros, upgrades, useful utility programs including a CMOS RAM editor plus much, much more. Can you afford to be without it?

Floating Point Assembler: A Dabhand Guide by David Spencer

Publication : August 1987. 250 pages

Book: £12.95. Disc: DFS 5.25in £7.95 ADPS 3.5in £9.95. Book and disc together £17.95 (ADPS £19.95)

The first ever BBC programming tutorial on floating point arithmetic in machine code. Learn the impossible - step by step details on floating point addition, subtraction, multiplication, division, sines, cosines - in fact everything. Lots of examples including multiple and double precision arithmetic. Another Dabs first!

If you would like more information about Dabs Press books and software then drop us a line at 76 Gardner Road, Prestwich Manchester M29 9FB, and we'll despatch our latest Dabhand Guides Guide.

Index

Using the Index

The index is mostly cross referenced so you should be able to find what you want at several points. For example to find a page number containing information on the bottom margin look under B for bottom or under M for margin.

Commands are shown in full uppercase characters and * commands can be found minus their *. So to find *CDIR look under C.

Page numbers are given normally however the odd letter does occur and this refers to the appropriate appendix.

Not every single occurrence of an item is index only the pages containing the hard facts. Thus there is no single entry for VIEW as this is mentioned on just about every single page.

Indexes are a pain to compile, but I hope this one is of use to you the reader. Me I'm off to Denmark for the weekend now so bye bye.
Regards Bruce.

abbreviations	116
access	37
Acorn User	13
adding words	151
ADFS	34,35
Advanced Disc Filing System	35
American character set	103
applications menu	16
asterisk	42
BBC	15
BBC B	18,23
block delete	23,47
block move	23,42
BM stored command	74
boot file	40,55,80,103,119,148
bottom margin	74
bottom of text	48
BREAK key	26
breaks, page	82

bytes free	18,I
caps lock	21,118
case swap	45
Cassette Filing System	34
cassette recorder	13
CAT	35
CDIR	35
centre text	66
CE stored command	66
CH	46
CHANGE	58,59,85
changing colour	118
changing rulers	28
chapter register	88
chapter sub-headings	89
character delete	24,45,48
character insert	24
character sets	103
characters maximum on line	87
character tab	31
CLEAR	45,48
CO	125
colour	118
columns tabs	30
command	35
command abbreviations	116
command delete stored	65
command star	33,117
command stored	22,65
comment line	125
CONFIGURE	117,123
constant VIEW	16,H
continuous stationary	32
control codes, printer	94
conventions	8
copying text	44
COUNT	25,50
counting with markers	50
CTRL-f0	23,47
CTRL-f1	62
CTRL-f2	27,53
CTRL-f3	52

CTRL-f4	51
CTRL-f5	28
CTRL key	22,24
cursor key	22,30
cursor keys simulation	24
Dabhand Guide	8,M
daisywheel printer	14,92
DATA	129
date register	88
default page	74
default printer	17
default ruler	28,31
define footer	76
define header	78
define macro	85
defining function keys	121
delete	115
delete block	23,47
delete character	24,45,48
delete command	65
delete end of line	46
deleting text	24,46
delete line	23,44,46,56
delete to beginning of line	123
delete up to character	46
delete word	123
DFS	34,35
DF stored command	76
DH stored command	78
dictionaries user	150
dictionary	141
dip switch	33
DIR	41
disc commands	39
disc drive	13,14
Disc Filing System	35
Disc Filing System, Advanced	35
disc labelling	38
DM stored command	85
document	12
documents long	113
dot-matrix printer	14,92

double line spacing	68
double sized print	98
driver, printer	19,93
Econet	13
EDIT	113
edit command	65,86
editing	12
editing no file	19
editing no file message	36
editing ruler	29
edit input file	114
edit mode	17,20,28
edit output file	114
elite sized print	98
emphasised print	94
emphasised	14
EM stored command	85
end macros	85
English character set	103
Epson	14,17,93
Epson printer	124
Epson printer codes	J
EP stored command	82
equipment	13
ESCAPE key	21
even page eject	82
ExCat	125
extended highlights	100
extending VIEW Manager	131
extkeys listing	120
f0	23,28,48
f1	48
f3	46
f6	23,44
f7	23,46
f8	24
f9	24,45
PCHECK	170
FI	20,22
file checker program	170
filename choice	37
filename conventions	38

files locking	37
FINISH	115
FJ	20,22
FM stored command	74
FOLD	58
footer	74
footer define	76
footer on and off	78
footer position of	76,79
footers	76
forced page eject	81
format	54,57
format mode	53
format paragraph	28,29
formatted text	27
formatting	51,55
formatting off	27
formatting text	102
FO stored command	78
FREE	113
free bytes	18
free bytes	I
function key editing	121
function keys	23
function keys programming	118
FX commands	117
goto marker	48
graphics dumps	14
hat wildcard character	62
header	74
header define	78
header margin	74
header on and off	78
header position of	78
headers	76,77
headings section	90
header position	79
HELP	15,171
HE stored command	78
highlight code guide	C
highlights	99,199
highlight matcher program	170

highlights extended	100
HILITE	171
hints and tips	116
HM stored command	74
HomePDG program	96
HT stored command	101
increasing bytes free	1
incrementing registers	88
indexes	153
information status	17,18
insert character	24
insert line	23,44,56
insert marker	43
insert mode	51
justification	12,51,52
KEEP	195
KEEP files	195
keyboard repeat rate	118
key repeat rate	118
labelling discs	38
laser printer	14
layout page	74
length page	74,79
letter	12
letter personalised	12
line count register	88
line delete	23,44,46,56
line delete to end of	46
line insert	23,44,56
line spacing	68
listing ExCat	132
listing File checker	170
listing Highlight matcher	170
listing Screen driver	162
listing VIEW Help ROM	171
listing VIEW Manager	134
listing VIEW recover	165
LM stored command	72
LOAD	80
loading printer driver	97
loading text	36
loading text to marker	49

locking files	37
long documents	113
long documents printing	115
LS stored command	68
LUNAR text	43
Master 128	15,18,23
macro names	85
macros	85,108
macros previewing with screen	86
macros printing	108
macros to number chapters	109
macros using them	109
margin	12,69,71
margin bottom	74
margin header	74
margins right	72
margin stored command	72
margin top	74
marked text saving	49
marker 1	42,44
marker 2	42,44
marker goto	48
marker insert	43
marker loading text to	49
marker not set	49
marker number	42
markers	42,49,54,148
markers 3 to 6	42,48
markers counting with	50
marker set	42
markers number	42
match next	62
Master Compact	13,14,18,23
MICROSPACE	102
microspacing	102
minus sign	42
MK	42,43,48
MODE	17,18,20,25,27,32
Mode screen	17,19,18
Mode shadow	18
monitor	13,14
MORE	115

MOUNT	16,35
move block	23,42
moving text	42
NAME	37
names macros	85
near letter quality	14,93
network	13,14
Network Filing System	34, 36
NEW	28,36,48
next match	62
NLQ	14,93
number registers	88
odd page eject	82
off	95
OLD	26
OP stored command	82
OPT	41
overtyp	51
OverVIEW	194
page default	74
paged mode	144
page eject forced	81
page ejecting	81
page layout	74
page length	74,79
page number setting	82
page register	82
page standard	80
pages two sided	83
paper sizes	165
paragraph format	29
paragraphs in VIEW	56
PB stored command	82
PE stored command	81
pica sized print	98
PL stored command	74,80
positioning footer	79
positioning header	79
pound signs	103
precis file	130
P register	82
previewing text	29

printing pounds	124
PRINT	101,158
PRINTER	13,97
printer codes	J
printer codes understanding	94
printer control codes	93
printer daisywheel	14,92
printer default	17
printer dipswitch	33
printer dot-matrix	14
printer driver	19,93
printer driver loading	97
printer effect codes	93
printer Epson	17
printer escape codes	94
printer parallel	117
printers and drivers	92
printers dot-matrix	92
printer serial	117
printer server	14
printers laser	14
printer type	17,92
printing long documents	115
printing macros	108
printing pounds	103
printing text	32
print multiple file printing	102
priority VIEW	H
programs disc	M
prompt VIEW	17
proof reading	193
quick save	37
QUIT	115
ragged edge	12
READ	49,54
register chapter	88
register date	88
register incrementing	88
register numbers	88
register P	82
register time	88
RENAME	58,115,158

REPLACE	58,61
register linecount	88
RETURN key	21
right justify	67
right margin	72
RJ stored command	67
ruler	20,27,28,53
ruler default	28,31
ruler editing	29
rulers changing	28
SAVE	34,35,36,49,169
saving marked text	49
SCREEN	29,40,65,68,82,99,115,163,194
screen driver stationary changes	164
screen layout driver	162
screen mode	17,19
screen preview	162
screen to preview macro	86
SEARCH	62,58,156
section headings	90
set marker	42
set register	82
setting page number	82
SETUP	54,124
shadow modes	18
SHEETS	32,92,99,101
SHIFT-f0	23,42
SHIFT-f1	45
SHIFT-f2	71
SHIFT-f3	46
SHIFT-f4	99
SHIFT-f5	99
SHIFT-f7	42,44,48,86
SHIFT-f8	65
SHIFT-f9	65
SHIFT key	23
SHIFT lock key	21
signs	103
sizes paper	165
spacing lines	68
SPELL	142
spell checker	141

SPOOL	194
SR stored command	82,88
standard page	80
star commands	117
stationary continuous	32
STATUS	124
status information	17,18
stored commands	22,65,74,78,101,B
style	192
sub-headings	89
subscript sized print	98
superscript mode print	98
swap case	45
switch dip	33
tabs	2,20,30, 31
table	2
tabulating figures	30
TAPE	34
television	13
templates	157
text beyond right margin	72
text centred	66
text copying	44
text deleting	46
text formatted	27
text formatting	51,55,102
text justification	51
text loading	36
text loading to marker	49
text LUNAR	43
text moving	42
text previewing	29
text printing	32
text right justify	67
text saving marked	49
text underlined	96
text wrap around	29
time register	88
TM stored command	74
top margin	74
top of text	48
TS stored command	83

TV	117
two sided pages	83
typing off screen	27
underlined text	96
underlining	14
understanding printer codes	94
unique words	143
unrecognised words	141
user guide	34
VIEW constant	16
VDUMP	166
versions	14
VIEW early versions	165
VIEW entering text	21
VIEW Help program	171
ViewIndex	155
VIEW Manager	125,126
VIEW paragraghs	56
VIEW priority	H
VIEW prompt	17
ViewSheet	131
ViewSpell	131,141
VIEW utilities	162
VIEW version changes	14, L
ViewIndex guide	G
Welcome disc	15
WIDE	195
wildcard character	62
WORD	15,16
word average length	18
word count	25
word delete	123
wordprocessor	11,12,13,18,25,34,42,58
words adding	151
words unrecognised	141
wrap around	29
WRITE	130

Notes

Notes











A Dabhand Guide

This is the most comprehensive tutorial and reference guide written about using the VIEW wordprocessor. Both the beginner and the more advanced user will find it to be an invaluable companion whether writing a simple letter or undertaking a thesis.

The author has left no stone unturned and all aspects of wordprocessing are covered. In addition a suite of VIEW utility programs are provided including, View Manager, an easily extendable front end. Thorny subjects such as macros, page layout and printer drivers are revealed in the authors own easy to follow style.

The many features of this book include:

- Compatible with BBC B and Master series
- Command and Edit Mode
- Rulers and Margins
- Markers to move, copy and delete
- SEARCH and REPLACE
- Stored Commands
- Page Layout
- Printer Drivers
- Macros and how to use them
- Hints and Tips
- View Manager
- Machine code utilities
- ViewSpell and ViewIndex

Bruce Smith is one of this country's leading authorities on the BBC Micro. Formerly Technical Editor of Acorn User he has written over a dozen books on the BBC Micro alone including the best selling The BBC Micro ROM Book, The Advanced Sideways RAM User Guide and Mastering Interpreters and Compilers.

£12.95

ISBN 1-87033-600-3



9 781870 336000